SECONDARY ION EMISSION FROM "SUPER-EFFICIENT" EVENTS: PROSPECTS FOR SURFACE MASS SPECTROMETRY

A Dissertation

by

RICHARD DALE RICKMAN

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2004

Major Subject: Chemistry

SECONDARY ION EMISSION FROM "SUPER-EFFICIENT" EVENTS: PROSPECTS FOR SURFACE MASS SPECTROMETRY

A Dissertation

by

RICHARD DALE RICKMAN

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Approved as to style and content by:

Emile A. Schweikert (Chair of Committee)	Marvin W. Rowe (Member)
Gyula Vigh (Member)	Ronnie Hart (Member)
Emile A. Schweikert (Head of Department)	

May 2004

Major Subject: Chemistry

ABSTRACT

Secondary Ion Emission from "Super-Efficient" Events: Prospects for Surface Mass Spectrometry. (May 2004)

Richard Dale Rickman, B. S. Arkansas State University

Chair of Advisory Committee: Dr. Emile A. Schweikert

Some collision cascades, induced by keV polyatomic projectiles, result in the emission of multiple secondary ions. Such co-emissions imply that the ejecta originate from molecules co-located within the nano-volume perturbed by a single projectile impact. The relevance for the chemical analysis of nano-domains depends on the effectiveness of the projectile to cause co-emission of two or more secondary ions. This research examines how projectile characteristics, i.e. the energy and number of constituent atoms in the projectile, influence multiple secondary ion emission, or "super-efficient" events. In addition we examine the relevance of this technique for nano-structure investigation.

Yields have been measured for multi-ion emission events as a function of projectile characteristics. The data show that some collision cascades are "super-efficient". For example, in a four-ion emission event, the yield for the phenylalanine quasi-molecular ion is two orders of magnitude larger from Au_4^+ impacts than from equal velocity Au^+ projectiles.

Yields for the co-emission of two phenylalanine quasi-molecular ions from "super-efficient" events have been measured. This case is particularly productive in that the detection of two analytically significant ions is recorded from a single event. Large increases (one to two orders of magnitude) in co-emitted ion yields were observed with increasing projectile energy and complexity. Correlation coefficients were calculated for the co-emission of two Ph ions, their behavior suggests differences in emission pathways for bombardment by atomic and polyatomic projectiles.

Finally, we use this methodology to investigate surface structural effects on the occurrence of "super-efficient" events. The results indicate that it is possible to distinguish between two phases of a chemical compound although the stoichiometry remains the same.

These results confirm previous predictions concerning the chemical nature of these "super-efficient" events. Also shown is that they are sensitive to the surface nano-environment. This approach extends the technology of Secondary Ion Mass Spectrometry by providing a methodology for probing surface nano-domains at the sub-100 nm level.

DEDICATION

To

my parents,

and

Linda, Jake, Zack,

and last, but most assuredly not least,

Claire

ACKNOWLEDGEMENTS

I would like to take this opportunity to thank the people who helped me over the last five years. My sincerest thanks go out to my mentor, Dr. Schweikert. At times it must have seemed I was trying to wear his patience out but, try as I may, I never could. He has been the source of seemingly endless knowledge and encouragement over the last five years. I would also like to thank Dr. Verkhoturov for his insightful comments and advice. Thanks Stan, for introducing me to some of the culinary delights the former Soviet Union has to offer, such as St. Petersburg chocolate, white cheese, and small fish packed in oil. Thanks also to Dennis James. I was not moving those computers around just to keep you on your toes, I promise. Mike Raulerson, too bad we never had time to play golf; I was just itching to show you how bad I am. Good luck with all you do, Linda and I will always keep your family in our minds. Finally, to the last of the authority figures, Charlene. It took me five years to figure out I was pronouncing your name wrong, please forgive me. Thank you for helping to decipher Dr. Schweikert's writing. I did not think anybody could write worse than me. Thanks also for your patience in walking me through the many mundane tasks and for always going the extra mile to make sure I was reimbursed for trips or settling other monetary issues.

I do not think graduate school would have been half as much fun if it were not for George. In between the many disagreements (I was, of course, always right) we got in several good rounds of golf. He taught me the finer things in life, like how to smoke a brisket or dress a deer. I also found out from him that you can use a university car for

such things as visiting “questionable” establishments or climbing the Rocky Mountains on a dirt road. Thanks also to Sara Balderas for her constructive comments concerning the TME. It is much more user friendly because of them. Good luck to you and George. I’m sure (mostly) that he will be a good husband. To Jay Locklear, may you one day be able to answer a yes or no question with yes or no. Good luck with C_{60}^{+} and finding some use for the laser you bought. Be careful with the micro-channel plates and try to make a CFD port last longer than two acquisitions. To Zhen Li, may China one day liberate Taiwan. Thanks for your technical expertise in matters pertaining to LCD projectors and laptop computers. You were more help than you know.

Thanks finally to my family. Linda stuck by me through the hard times as well as the good (it seemed that the former outnumbered the latter). You have been the greatest source of inspiration for me. All that I did was for you. Thanks for putting up with the long hours I worked and the times I was away. I am lucky to have you. A lesser person would have given up a long time ago. Russ, at least in our case, was wrong. Finally, to Jake, Zack, and Claire, you three have been, and will always be, the accomplishments that I am the proudest of.

TABLE OF CONTENTS

CHAPTER Page

LIST OF FIGURES

FIGURE Page

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

Surface analysis is a key issue in many areas of science and technology. There are a number of techniques which can provide information on atomic and/or molecular species present at the surface but none offers the scope of information, from isotopic to molecular and from trace to imaging, as does Secondary Ion Mass Spectrometry, SIMS (Fig. 1-1) [1]. In addition, this technique, described in some detail in Chapter II, probes only the topmost monolayers of the surface (Fig. 1-2) [1].

SIMS is widely used to detect and often localize analytes in biological, geological, environmental, semiconductors, and both organic and inorganic materials [2-7]. The performance of SIMS depends on the secondary ion yield. For atomic projectiles such as Cs^+, O^+ or Ar^+, with 3 to 20 keV impact energies, secondary ion yields typically range from a few hundredths of a percent to a few percent [8]. Secondary ion yields can be boosted close to unity using laser post-ionization for elemental species and small organic molecules [9, 10]. For molecular species this approach is hampered by photo-fragmentation and/or limitations imposed by specific ionization schemes. Enhanced secondary ion yields can be also be obtained by substituting polyatomic projectiles for atomic ones. For example, Szymczak et al. measured yield enhancements, for molecular secondary ions, of ~100 times for SF_5^-,

---

This dissertation follows the style and format of Physical Review Letters.

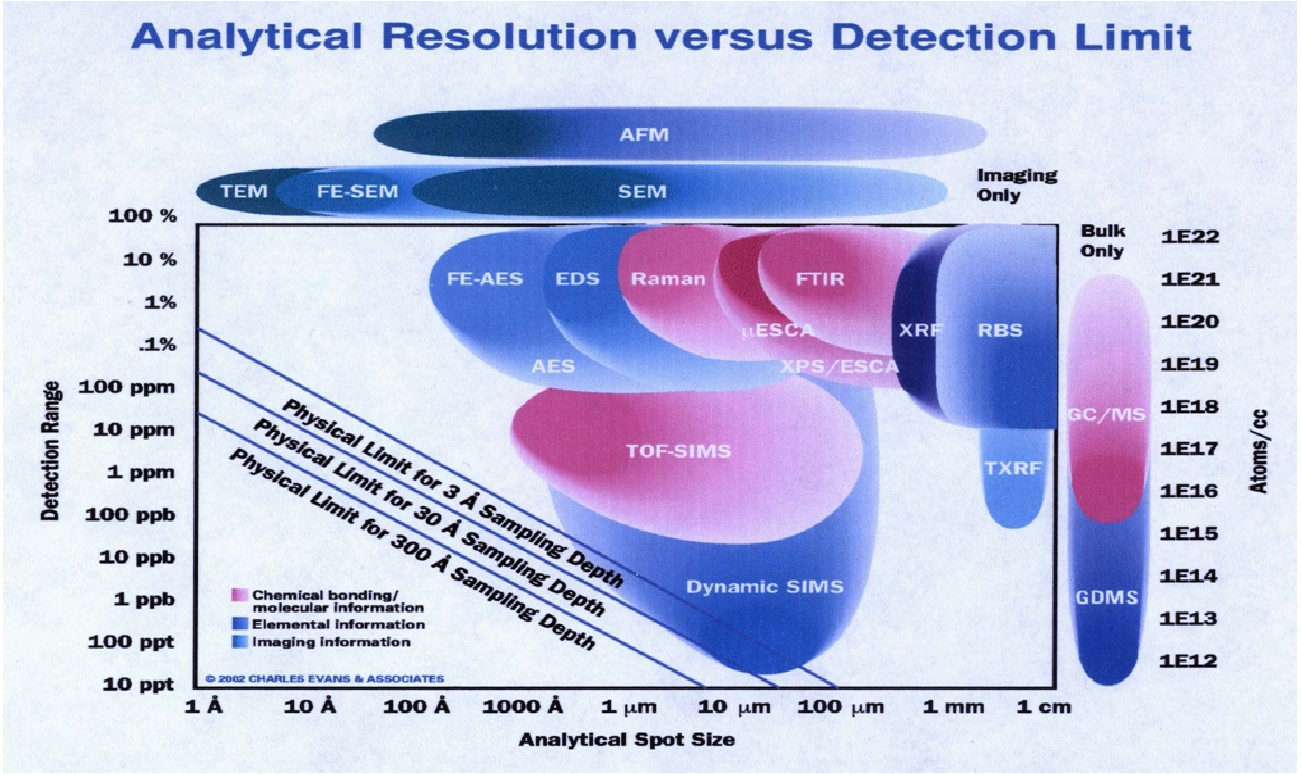


Fig. 1-1: Sensitivity comparison of surface analytical techniques. *Reprinted with permission from Ref. [1].*

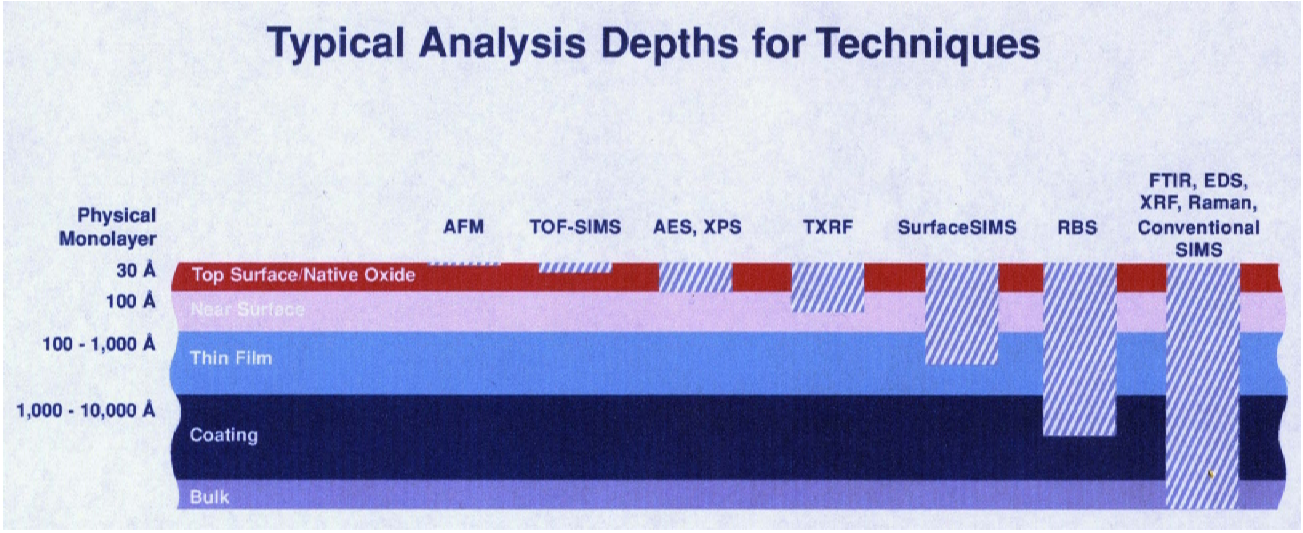


Fig. 1-2: Sampling depth comparison of various surface analytical techniques. *Reprinted with permission from Ref. [1].*

SF_5^+, and SF_6^- projectiles compared to Xe^+ projectiles[11].

The number of secondary ions ejected from a single projectile impact varies. In most cases an event produces no secondary ions, the next most probable occurrence is the emission of only one secondary ion. There are, however some "super-efficient" events in which multiple secondary ions are formed. That cluster bombardment can lead to enhanced secondary ion yields is well known. It has been postulated that these enhancements result primarily from an increase in the frequency of "super-efficient" events.

An additional feature is that co-emitted secondary ions originate from a nanometric volume [12]. Molecular dynamics simulations of these impacts indicate that desorption occurs from an area as small as 10 nm in diameter [13]. Thus, if two or more ions are desorbed from the impact of a single projectile, they are spatially related on this scale. The feasibility of probing co-located surface species, via the detection of two coincidentally emitted secondary ions, has been demonstrated on polymer blends [14] and silver nanoparticles [12]. A prerequisite for nanoanalysis via coincidental ion, CI, emission is to have bombardment conditions conducive to multi-ion emission. However, the data on single secondary ion yields (the number of ions of a specific mass emitted per projectile) show that most projectiles impacts are null events. Thus, folded into many null events, are a few productive ones yielding one or more secondary ions. The question then arises, do the latter involve "super-efficient' collision cascades which differ from those leading to the emission of single secondary ions?

The occurrence of "super-efficient" collision cascades raises two questions. First, what is their rate of production as a function of projectile characteristics? Further one must wonder if co-emissions are due to different pathways in desorption/ionization with respect to single ion emission or simply due to random occurrences.

From the analytical chemist's viewpoint, coincidental ion emission is only of interest if the ejecta are accurately reflect the physical and chemical environment of the nano-domain. It has been shown earlier that a compound probed successively in a gel and in a crystalline form can exhibit significant differences in the corresponding SIMS spectra albeit there is no change in the compounds stoichiometry [15]. We report below on the further development of a SIMS methodology pertinent for probing surface structure.

To explore the issues raised here the first task of this research was to construct a time-of-flight secondary ion mass spectrometer with a cluster ion source which would provide a selection of polyatomic projectiles. A key concern in the instrument design was that experiments could be conducted in the event-by-event bombardment/detection mode. The second task of this research was to develop software capable of collecting and organizing raw experimental data, in the event-by-event mode.

With the aims outlined above, the first objective was to measure secondary ion yields from specific types of ion emission events, i.e. one ion emission events, two ion emission events, etc. as a function of the projectile type and its kinetic energy.

The second objective of this study was to measure coincidental secondary ion yields and correlations from a model organic compound as a function of different projectile characteristics, i.e. the type of projectile and its kinetic energy.

The third objective was to evaluate the utility of cluster SIMS for the physical characterization of nano-domains. The test case focuses on collecting SIMS data from amorphous and crystalline materials, each having the same stoichiometry, then looking for differences in the mass spectra and in the secondary ion yields from specific types of ion emission events.

CHAPTER II

SECONDARY ION EMISSION FROM keV ATOMIC AND POLYATOMIC PROJECTILE IMPACTS

The purpose of this review is to summarize developments in Secondary Ion Mass Spectrometry pertinent to the present study. The technique is discussed along with fundamentals of ion-induced secondary ion sputtering, secondary ion yield enhancements, secondary ion multiplicities (defined later), and the use of SIMS for surface structural characterization.

Secondary Ion Mass Spectrometry

Secondary Ion Mass Spectrometry, SIMS, is an analytical technique whereby a projectile with a few hundred eV to 10's of keV is used to probe a target surface. The analytical signal comes from the sputtered and ionized secondary ions which are mass analyzed by any of a number of techniques, e.g. Time of Flight, ToF, quadrupole mass analyzers, and both magnetic and electric sector mass analyzers. The sputtered secondary ions originate from the topmost atomic layers of the sample and thus make this technique one of the most surface sensitive of all surface analytical methods [16]. Today SIMS enjoys a wide variety of applications. SIMS is widely used for analysis of trace elements in solid materials. A SIMS primary ion beam can be focused to less than 1 um in diameter. Rastering the primary ion beam across a sample surface provides for microanalysis giving the lateral distribution of elements on a microscopic scale. A

SIMS analysis can be categorized by the primary ion fluence used to probe the surface. Dynamic SIMS analysis employs high primary ion fluences, ($> 10^{12}$ primary ion/cm^2) consequently the sample surface is slowly sputtered away. Secondary ion signals are continuously analyzed during the sputtering process, providing chemical information as a function of depth, i.e. a depth profile. If the primary ion fluence is low ($< 10^{12}$ primary ions/cm^2) the technique can be considered virtually non-destructive. This is what is known as static SIMS. The static regime minimizes the damage done to surface species and the resulting ion fragmentation patterns contain information useful for identifying molecular species.

Secondary ion mass spectrometers can be grouped into two categories: imaging and non-imaging instruments. The first commercially available imaging instrument, or ion microscope, was produced by Cameca S. A. in 1965 and was based on a design by Castaing and Slodzian [17]. This instrument permitted microanalysis of a solid surface with lateral resolutions on the order of one micron. Variations of this microscope are still in production today with lateral resolutions on the order of 50 nm. These can be operated in both the ion microscope mode or as a scanning ion microprobe (the primary ion beam is scanned across the sample while simultaneously recording the secondary ion signal). The first commercial non-imaging instrument was based on the design of Herzog and Viehboeck [18] and was introduced in 1967. This instrument consisted of a primary ion source, secondary ions were analyzed with a double focusing mass spectrometer. This instrument was used primarily for trace analysis (sensitivity in the low to sub ppm range), elemental depth profiling, and analysis of thin films.

Early SIMS instruments employed primary ion sources that generated atomic projectiles. Typical secondary ion yields are a fraction of a percent when atomic projectiles are employed. As mentioned in Chapter I, polyatomic projectiles can increase the sensitivity of a SIMS measurement by increasing secondary ion yields. Today there are commercially available cluster primary ion sources which can generate a number of cluster projectiles e.g. C_{60}^+ or Au_n^+ ($2<n<5$).

Fundamental Theories

A prerequisite for successful interpretation of secondary ion mass spectra is a fundamental understanding of sputtering processes. The first theory that adequately explained the sputtering process was developed by Sigmund in 1969 [19]. In this model target atoms are set into motion by elastic collisions with the projectile. The target atoms themselves are free now to undergo elastic collisions with neighboring atoms. This process leads to the development of a collision cascade. If the target atom is directed upward it has some probability of escaping the surface either as an ion or a neutral atom. This theory described sputtering yields for low mass projectile atoms, but failed to describe sputter yields for both heavy and polyatomic projectiles. One reason for this was the basic assumption that moving atoms will always collide with atoms at rest, resulting in what is termed a linear collision cascade. To account for the failure of this model the concept of a thermal spike, or a region of high energy density within the target was introduced [20, 21]. In this model the collision cascade is of sufficient density that essentially all atoms in the volume are set into motion. This model could

describe experimental observations not predicted by the linear cascade model e.g. enhanced sputter yields.

The physical mechanism by which the spike transfers its energy to the surface, resulting in enhanced sputtering yields is the subject of some debate. One mechanism is that a shockwave propagates through the material and, as it intersects the surface, some secondary particles are ejected [22]. Thompson *et al* argued that mesoscopic roughening of the surface, or radiation induced damage resulting in a lowered surface potential, could lead to the observed enhanced sputtering yields [23]. Regardless of the mechanism the spike models were able to reasonably predict sputtering yields, each having varying degrees of success. The idea that an area of high energy density was necessary for enhanced sputtering yields was established [24, 25]. From an analytical standpoint, enhanced sputter yields translates to enhanced sensitivity. The task was to find projectiles that are more efficient at producing conditions that lead to enhanced secondary ion yields.

Secondary Ion Yield Enhancements

One approach for comparing yields of atomic and polyatomic projectiles is to calculate the enhancement factor, ε. A keV polyatomic projectile will break apart into its constituent atoms, n, as it strikes a surface. This is due to the fact that the energy that binds the cluster (~ 1 – 10 eV) is much less than the overall kinetic energy, E, of the projectile. Each individual atom will retain a proportional amount of the cluster projectile's original kinetic energy. The individual atoms are now free to initiate their

own cascade leading to the desorption and ionization of secondary particles. For this reason, measurement of ε is made at the same *E/n* for the projectiles that are to be compared. The enhancement factor is defined as follows:

$$\varepsilon = \frac{Y_m n}{Y_n m} \quad (\text{for } m > n) \qquad \textbf{Eq. 2-1}$$

where Y_m is the secondary ion yield from an *m*-atom projectile and Y_n is the secondary ion yield from an *n*-atom projectile. For values of $\varepsilon > 1$ there is an enhancement in secondary ion yields for the *m*-atom projectile.

The first observation of enhanced sputtering from polyatomic ion bombardment was as early as 1960's [26, 27]. Up until 1987, studies with polyatomic projectiles were done on poly-crystalline metal targets and their oxides with the objective of elucidating fundamental sputtering mechanisms. The application of polyatomic projectiles for organic sample analysis by Appelhans et al. in 1987 demonstrated that these types of projectiles could offer orders of magnitude increases in secondary ion yields when compared to atomic projectiles [28]. The scope of this research was broadened in 1989 to investigate additional projectiles [29]. In the latter study, the damage cross section and ion formation efficiency, in addition to secondary ion yields, were measured as a function of projectile characteristics. Seminal studies of yield enhancements via polyatomic projectiles, sometimes termed the "cluster effect" or supra-linear enhancements, were carried out in the late 80's, and continue today, by groups at Texas A&M and the Institute de Physique Nucleaire at Orsay. In 1989 Blain et al. measured the secondary ion yields of the phenylalanine molecular ion from Cs^+, $(CsI)Cs^+$, and

$(CsI)_2Cs^+$ [30]. Values of ε up to 50 were measured and, within the energy regime studied, it was found that the secondary ion yields were proportional to the square of the momentum of the projectile. Blain *et al*.'s study was extended in 1991 to include secondary ion yields from Au_n^{m+} (n = 1-5, m = 1-2) bombardment of phenylalanine targets [31]. In addition to measurements of molecular secondary ion yields, yields were calculated for atomic secondary ions, e.g H^-. It was determined that the number of constituent atoms in the projectile is an important parameter effecting secondary ion yields. Additionally noted was that supra-linear enhancements were more prevalent in complex secondary ion yields than in atomic secondary ion yields.

Secondary Ion Multiplicities

In addition to enhancing secondary ion yields, polyatomic projectiles have been shown to increase the average secondary ion multiplicity. In this case ion multiplicity is defined as the number of secondary ions (of different mass) detected from a single projectile impact. Zubarev et al. measured these distributions for a variety of projectiles at different energies [32]. The main experimental result from this study was that, within the energy range explored, the average multiplicity increased with increasing projectile velocity and with increasing mass for projectiles having the same velocity. A higher average multiplicity can be advantageous from an analytical viewpoint. Having more secondary ions detected from a single projectile impact increases the amount of chemical information received from the nano-domain probed by the projectile.

Surface Structure Characterization

Physical characterization of various analytes via SIMS has been limited mainly to depth profiling of dopants implanted into semi-conductor wafers (nm depth resolution) or ion mapping of a target using imaging SIMS (~100 nm resolution). Provided the SIMS experiment is conducted in the event-by-event bombardment/detection mode, it is possible to probe only the area perturbed by a single projectile. This makes this a powerful technique for both chemical and physical characterization of nano-domains. However, little attention has been given as to what extent the SIMS signal can provide information related to surface structure. Using Plasma Desorption Mass Spectrometry, PDMS, (in this case secondary ions are generated from the impact of a ^{252}Cf fission fragment having energies in the MeV range), Van Stipdonk et al. demonstrated that secondary ions related to the original stoichiometry and structure of the analyte could be obtained from the negative ion spectra [15]. Van Stipdonk et al. observed both qualitative and quantitative differences in the mass spectra for both amorphous and crystalline compounds of ZrP. To date no similar studies have been carried out using SIMS.

The utility of SIMS for nano-object characterization will require projectiles that maximize the amount of chemical information received from single projectile impacts. However the issue is not simply to increase the frequency of these "super-efficient" events, the key condition that must be met is that the detected secondary ions reflect the chemical conditions that exist on the surface.

CHAPTER III

INSTRUMENTATION AND ELECTRONICS

Instrumental

A schematic of the Cluster SIMS instrument used for the experiments described in Chapters V - VII is given in Figure 3-1. There are two principal regions: the source, or primary ion leg, and the Time-of-Flight, ToF, or secondary ion leg.

Time of Flight Region

The ToF region is housed in a 29.5 cm diameter, custom designed, stainless steel chamber (Kurt J. Lesker, Clairton, PA) with a base pressure of ~10^{-7} torr. Vacuum conditions are maintained by a 760 l/s diffusion pump (Edwards Vacuum Products) backed by a 12 cfm rotary vane mechanical pump.

The sample is inserted into a holder, constructed of Teflon©, via the sample introduction system. This introduction system has been incorporated to facilitate sample change while maintaining vacuum in the main chamber. The sample is attached to a threaded rod. The rod and sample are then attached to the sample port via a KF© fitting and the region is evacuated by a mechanical pump. Once this region is evacuated (< 10^{-3} torr) a gate valve is opened to the main chamber and the sample is inserted. The time it takes to insert the sample and reach conditions in the main chamber adequate to apply detector bias (<10^{-6} torr) is 3 to 4 minutes making this an efficient method for high throughput SIMS analysis.

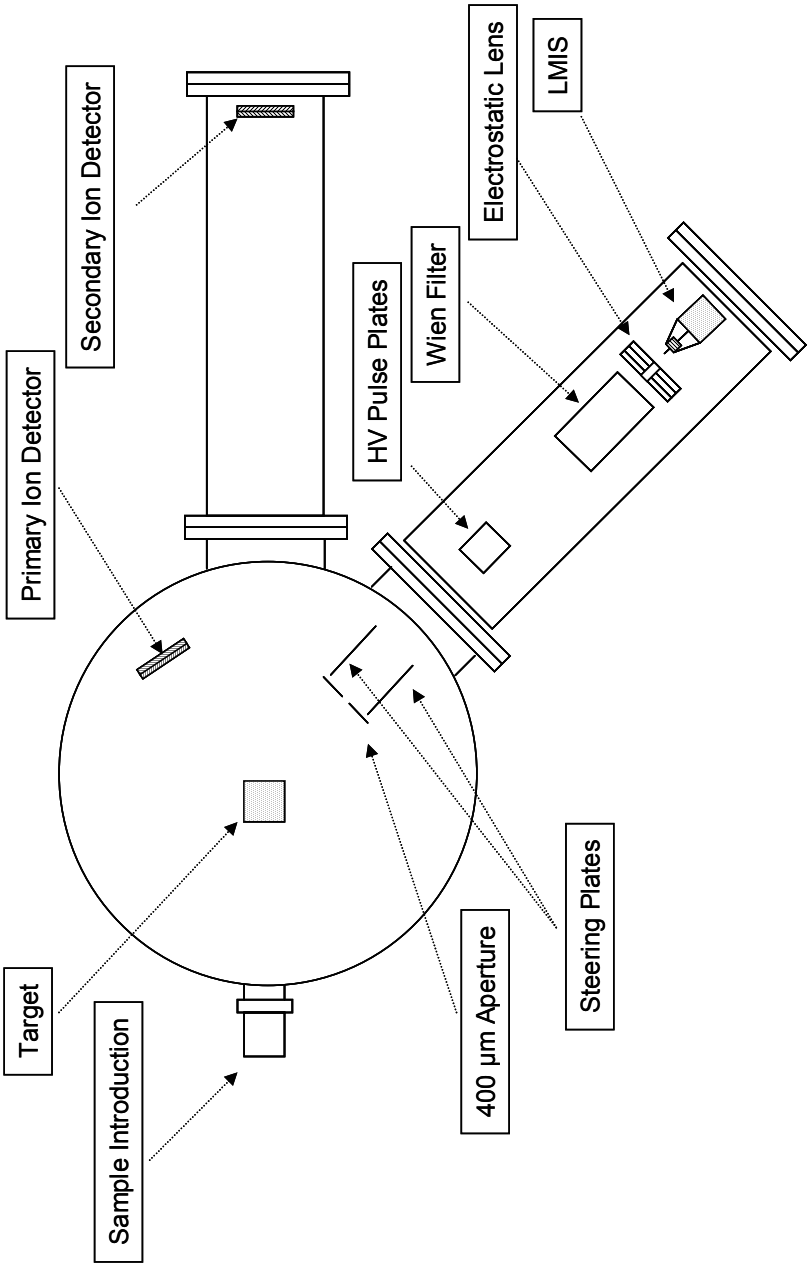


Fig. 3-1: Schematic of cluster time-of-flight secondary ion mass spectrometer.

The sample cube is made from stainless steel or brass with dimensions of 1.5 cm x 1.0 cm x 1 .5 cm. The cube is biased and separated by 5 mm from a grounded 90% transmission grid (Buckbee-Mears, St. Paul, MN). This provides the potential field gradient necessary to accelerate secondary ions and electrons towards their respective detectors. Secondary electrons are emitted when the primary ion strikes the target. These secondary electrons (whose flight times are negligible compared to that of secondary ions) are steered by a weak magnetic field towards the primary ion micro-channel plate, MCP, detector assembly. Their arrival gives the start signal for ToF mass analysis of any secondary ions ejected by this primary ion. The secondary ions are accelerated into a field free region where they will separate, in time, based on the square root of their mass to charge ratios. The field free region is approximately 56 cm long with a calculated transmission efficiency of nearly 90%. The sample holder and primary ion MCP assembly are attached to an adjustable platform secured by four screws to the interior of the vacuum chamber. Alignment of this platform, and thus the target, with the secondary ion detector is accomplished, visually, by adjusting the platform while looking through the sample introduction port, down the field free region, to the secondary ion detector. This alignment must be done while the system is not under vacuum. This optimization is necessary anytime the platform is removed for maintenance, i.e. changing grids or MCP detector maintenance.

Source Region

The primary ion leg is approximately one meter long and extends from the gold Liquid Metal Ion Source, Au-LMIS, (Institute de Physique Nucleaire, Orsay) to the target, Figure 3-1. The ion source is housed in a custom designed chamber with the pressure maintained at ~ 10^{-7} torr. Vacuum is maintained by a 60 l/s turbo-molecular pump (Pfeiffer Vacuum Technologies Inc.) backed by an 1.2 cfm rotary vane mechanical pump. The source and ToF regions are separated by gate valve which allows for sample change, or adjustments in the ToF region, while the source is energized.

The emission from the Au-LMIS consists of Au_n^{m+} ions ($1 \leq n \leq 10$; $1 \leq m \leq 2$). Instructions for fabrication are in Appendiix A. The emission current from the singly charged atomic projectile is ~ 2 nA (measured at the target). Currents on the order of 10 – 500 pA are typical for the remaining ions. The ions are extracted and focused using a three electrode electrostatic lens (Fig. 3-2). Adjustments of the potentials on these lenses are required in the event the kinetic energy, K_e, of the primary ion is changed. Provided U_3 is kept at ground, the potentials on U_1 and U_3 are proportional to the K_e, of the primary ion. A plot of this relationship is given in Figure 3.3 for the two adjustable potentials, U_1 and U_3 (Fig. 3-2). An equation from the linear regression of each of these lines can be solved for V (in kV) provided the K_e, of the primary ion is known. This gives reproducible settings for a range of primary ion energies.

The focused beam is mass analyzed using a wien filter. A wien filter consists of an electric field perpendicularly crossed by a magnetic field. Ions having a certain velocity will pass through the filter, unaffected by the fields, while all others are

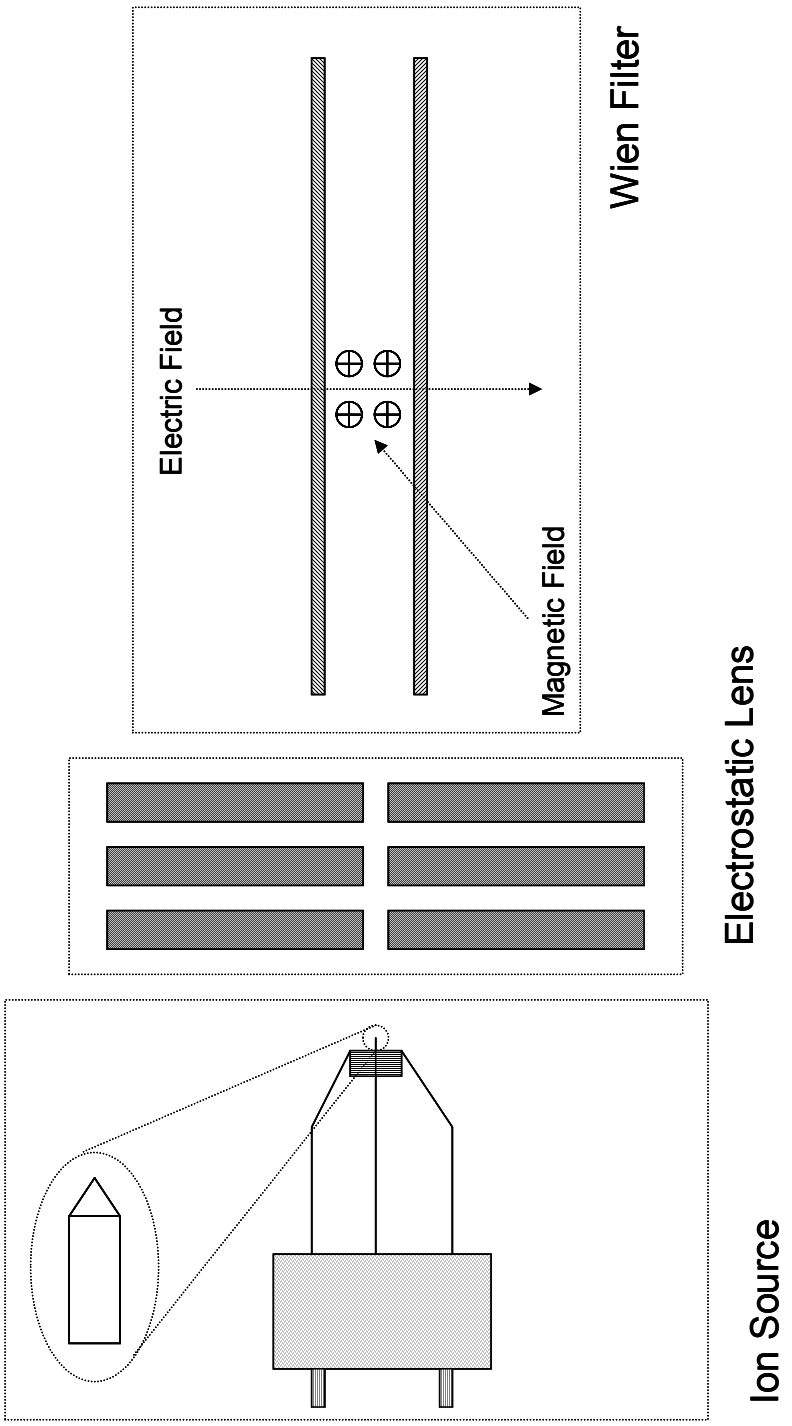


Fig. 3-2: Schematic of ion source, electrostatic lenses, and Wien filter for SYS VII.

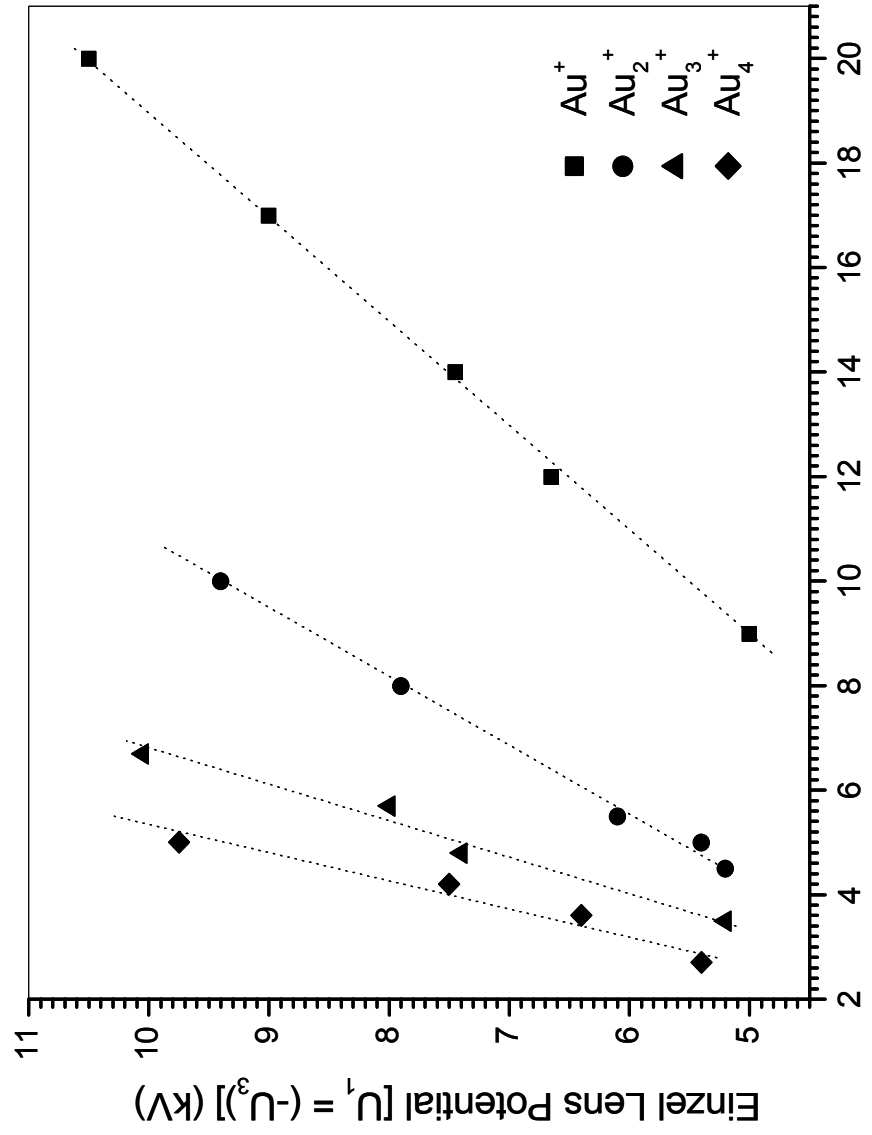


Fig. 3-3: Plot of the einzel lens potentials vs. the kinetic energy per atom for the selected projectiles.

deflected. Projectile selection is accomplished by varying the electric field while maintaining a constant magnetic field. Figure 3-4 is the primary ion spectrum from the unfiltered primary ion beam from the Liquid Metal Ion Gun. Once the wien filter is adjusted, a particular primary ion can be selected for use (Fig. 3-5). The electric field for the wien filter is provided by a 0-1kV power supply (Model 935 Quad 1 kV Power Supply, Ortec) with 0.1 V resolution in the 0 – 100V range and 1 V resolution in the 100 – 1kV range. The magnetic field for the wien filter is generated by an electromagnet (Electropreci, Paris, FR) powered by a constant current/constant voltage power supply (Bertan,). Settings for the wien filter will differ with projectile energy and mass. A simple equation can be derived [33] which relates the potential applied to the electric field to the energy and mass of the projectile:

$$V = k\sqrt{\frac{K_e}{M}} \qquad \textbf{Eq. 3-1}$$

Where V is the potential in volts (+/-) applied to the wien filter plates, k is an empirically derived constant, which includes the magnetic field strength and filter dimensions, M is the mass of the projectile in amu, K_e is in electron volts.

The mass filtered ion beam passes next through a set of high voltage pulse plates. One plate is held at ground while the other is switched from a positive to a negative potential. The switch is a push-pull MOSFET (Belkhe, HTS 151-03-GSM, Germany) capable of switching between +/- 15 kV at a sustained rate of 10 kHz. Pulse widths can be as narrow as 200 ns with rise times of ~ 25 ns. The high voltage is supplied to the switch by two 1 kV high voltage power supplies (Glassman, Series EK, High Bridge,

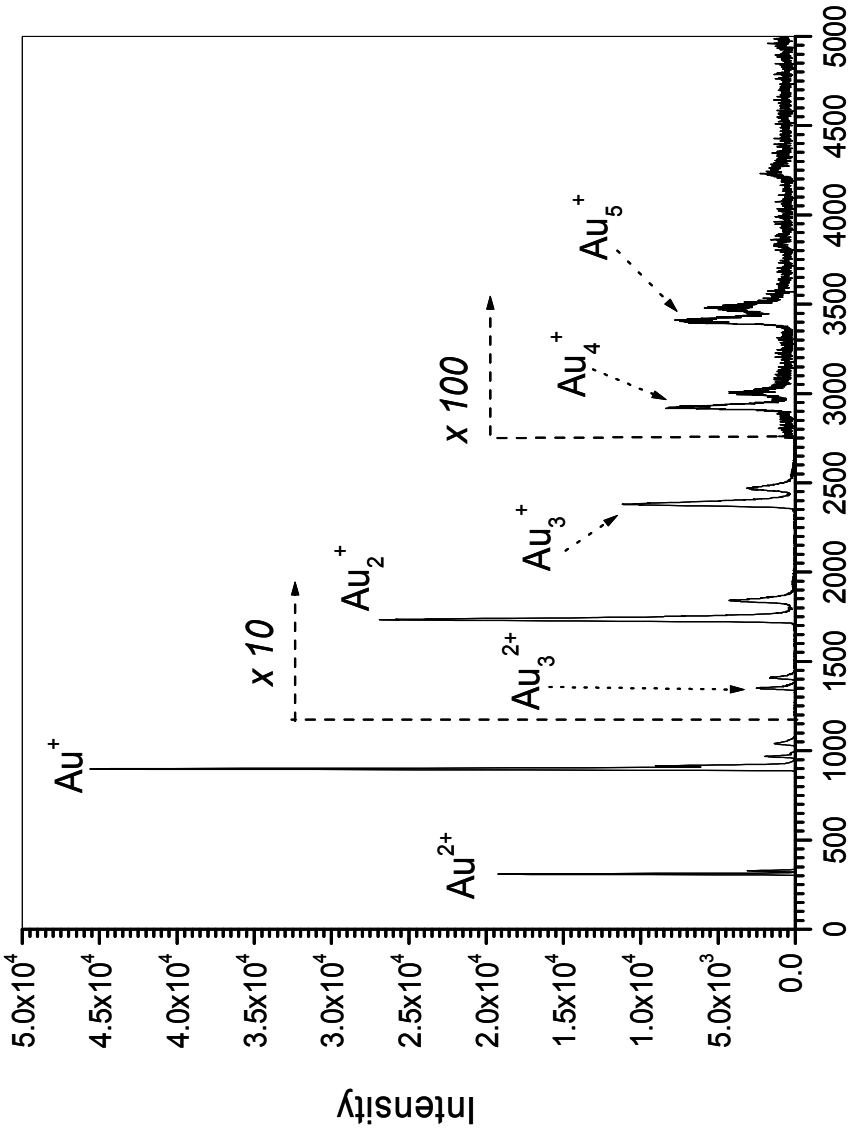


Fig. 3-4: Primary ion spectrum of LMIS emission current with Wien filter off.

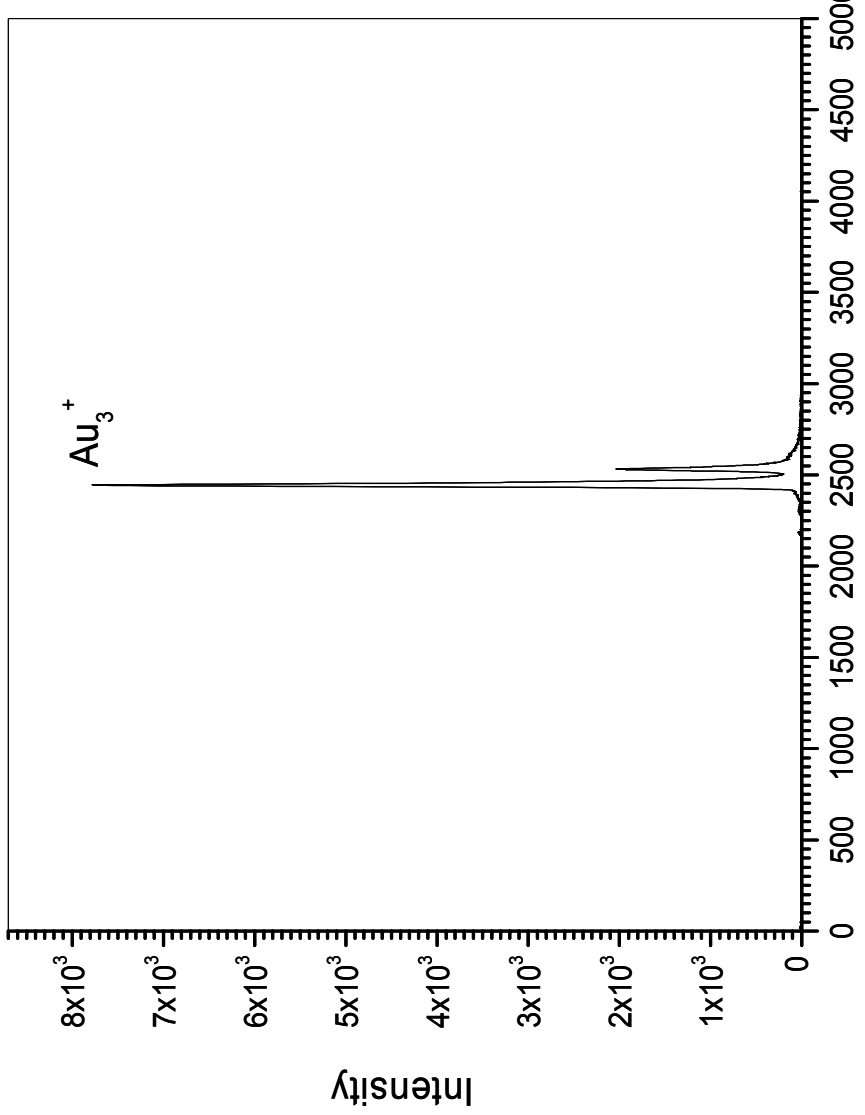


Fig. 3-5: Primary ion spectrum of LMIS emission current with Wien filter adjusted for Au_3^+.

NJ). The switch is triggered from the output of a pulse generator (Protek, Model B 1010).

Pulsing is necessary for two reasons. One is that it provides a start signal for the time-of-flight mass analysis of the primary ion beam (described later in the last section of this chapter). More importantly, it reduces the fluence of the primary ion beam. This type of pulsing is what is known as the Differential Impulse Sweep Method, DISM, [34]. Briefly, ions enter the pulse region and the potential is switched from positive to negative. Ions enter the region and the switch is triggered. Only those ions at a certain position will have the same trajectory both before and after the pulse. These can then continue to the target. The number of ions that will have their trajectories unaffected, and thus reach the target depends, primarily in our case, on the amplitude of the pulse and the ion beam current, either of which can be adjusted. The beam current can be reduced by defocusing using the potentials on U_1 and U_3 (Fig. 3-2). However, it is preferable to adjust the pulse amplitude. Adjustments in focusing may require readjustment of the wien filter settings. The primary ion(s) are swept across a 400 μm diameter collimator. The aperture, combined with pulsing, reduces the number of primary ions that reach the target to ~ 0.1 ion / pulse. This is adequate for making measurements in the event-by-event mode.

A set of vertical steering plates center the primary ion on the target. If the K_e of the primary ion is changed then the potential to the plates must also be changed. One can plot the potential applied to the plate as a function of the K_e of the primary ion

(steering plate potential is optimized in each case when a maximum secondary ion yield is realized). Figure 3-6 is such a plot. Knowing the primary ions K_e allows one to calculate the steering plate potential by solving the linear equation. It must be remembered that the mathematical expressions describing the relationship between the primary ion K_e and einzel lens potentials (Fig. 3-3), the wien Filter potential (Eq. 3-1), and the steering plate potential (Fig. 3-6) are derived from empirical data. Any adjustment of the sample platform, or whenever the source is removed for maintenance or replacement, requires reevaluation of these constants.

Principles of Time-of-Flight Mass Analysis

For all experiments described in this dissertation, ToF mass analysis is used for secondary ion identification. The advantage of ToF over other techniques, i.e. quadrupole mass filters, magnetic and electric sector instruments, etc., is that *all* ions from an event are sampled whereas in other techniques most of the ions, in many cases greater than 90%, go undetected [8]. ToF mass analysis is possible because temporal separation occurs in a field free drift region of ions having the same kinetic energy, but different masses and therefore different velocities.

From the time an ion is created until the time it is detected it will encounter at least two distinct regions that will affect its recorded flight time. The first region is the acceleration region (Fig. 3-7). The time it takes the ion to traverse this region, t_a, is given by:

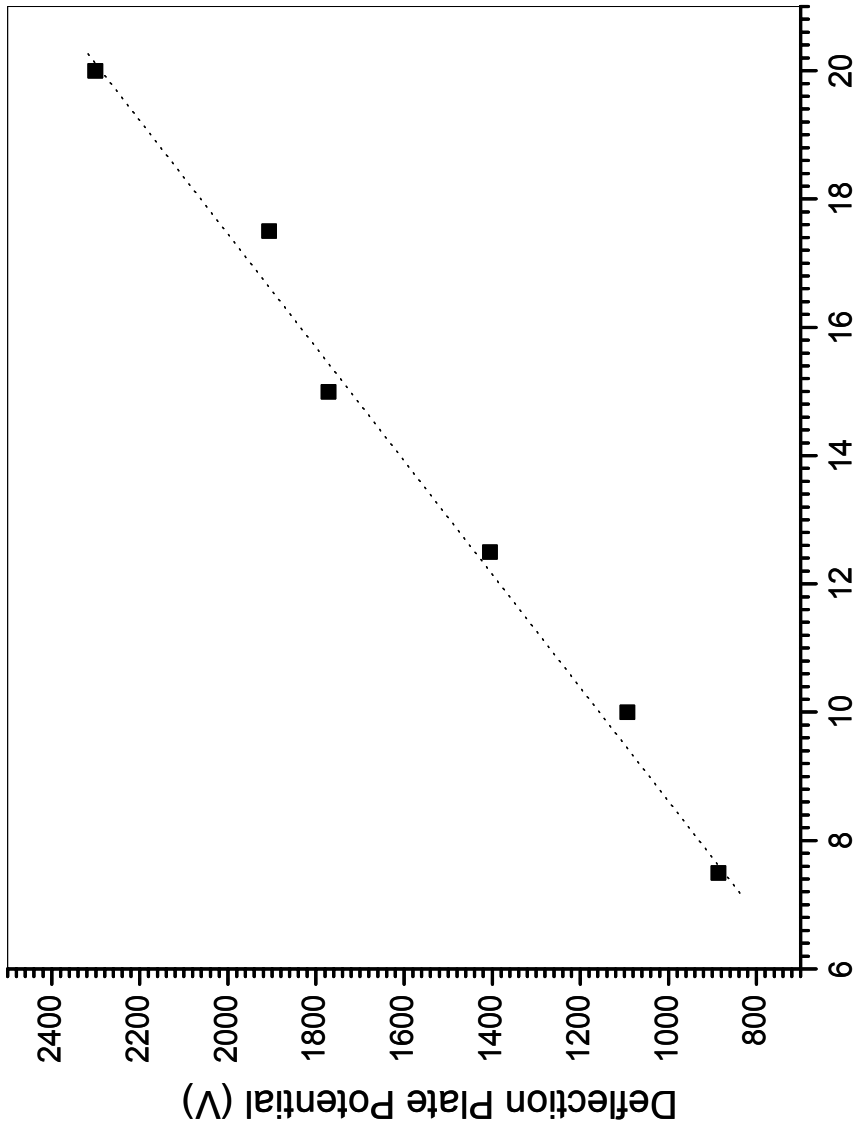


Fig. 3-6: Plot of deflection plate potential vs. kinetic energy of the primary ion.

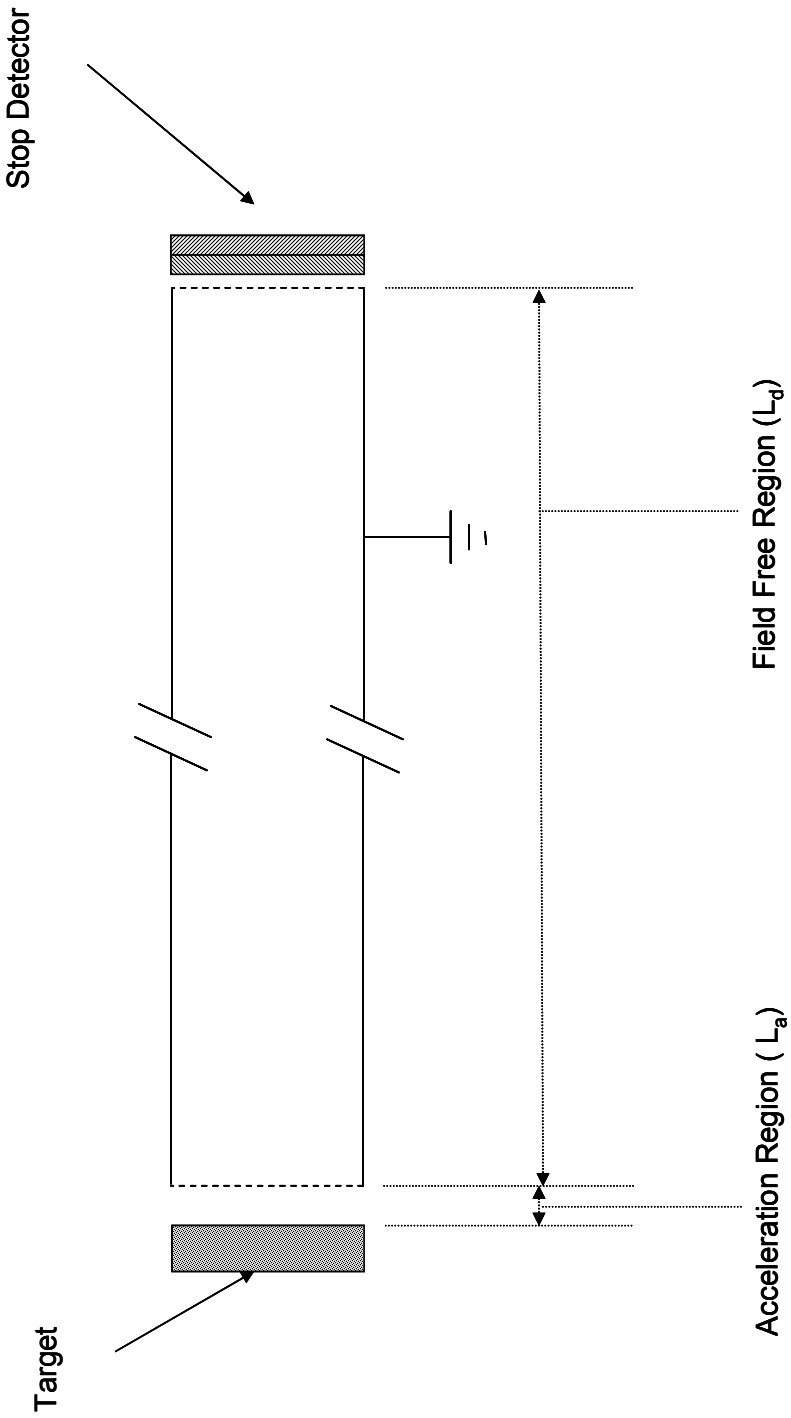


Fig. 3-7: Time-of-flight schematic.

$$t_a = \sqrt{\frac{2L_a^{\,2}}{V}}\sqrt{\frac{m}{q}} \qquad \textbf{Eq.3-2}$$

where L_a is the length of the acceleration region, V is the target bias; m is the mass of the ion with charge q. Once the ion is accelerated it will next encounter the drift or electric field free region. The time the ion spends in this region is given by:

$$t_d = \sqrt{\frac{L_d^{\,2}}{2V}}\sqrt{\frac{m}{q}} \qquad \textbf{Eq. 3-3}$$

where L_d is the length of the drift region. Thus the total flight time of the ion t_{tot} is:

$$t_{tot} = t_a + t_d = \sqrt{2}\sqrt{\frac{m}{q}}\left(\sqrt{\frac{L_a^{\,2}}{V}} + \frac{1}{2}\sqrt{\frac{L_d^{\,2}}{V}}\right) \qquad \textbf{Eq. 3-4}$$

Once the spectrum is collected it must be mass calibrated. Knowledge of L_d, L_a, V, and t_{tot} allows for mass identification using Eq. 3-4. This would entail a tedious calculation for the mass assignment of each peak. A much simpler approach is used. Knowing that the flight time, t_{tot}, of an ion is proportional to the square root of its m/q ratio, all other factors can be reduced to two constants giving the following [35, 36]:

$$t_{tot} = \sqrt{\frac{m}{q}}C_1 + C_2 \qquad \textbf{Eq. 3-5}$$

If the identities of at least two ions are known, then it is possible to solve a system of two equations for C_1 and C_2 to calibrate the entire spectrum. Typically H^- and C_2H^- are used due to their abundances and presence in all spectra. A more precise determination can be made through iterative determinations of C_1 and C_2. The ability to

resolve two different masses, *m* and *Δm,* is defined by their ratio. However the spectrum is initially in the time domain so a more expedient approach is to have this ratio in units of time. It can be shown that:

$$\frac{m}{\Delta m} = \frac{t}{2\Delta t} \qquad \textbf{Eq. 3-6}$$

or, because each channel number, *Ch#*, is proportional to time:

$$\frac{t}{2\Delta t} = \frac{Ch\#}{2\Delta Ch\#} \qquad \textbf{Eq. 3-7}$$

Micro-channel Plate Detectors

Secondary ion and electron detection was accomplished using micro-channel plate, MCP, detectors. MCP's are particularly well suited for ToF mass analysis due to their fast response, fast effective recovery times [37], and charged particle detection efficiencies [38]. The MCP's used in these experiments were ToF quality (Part # 30286, Burle Electro-Optics, Sturbridge, MA). The minimum active area is 25 mm in diameter. Channels are 10 μm in diameter with a bias angle of 12^o on 12 μm centers. This gives an open area ratio, or maximum detection efficiency of ~ 60%. The MCP detector consists of two MCP's arranged in a chevron array giving a typical gain, or number of electrons generated per charged particle impact , of ~ 10^6.

Figure 3-8 is a diagram of a typical chevron array MCP detector. The potential is divided by a network of resistors, R_1 and R_2 from ~ +2.2 kV on the anode to ground on the front plate. The high voltage input is conditioned by a high pass filter, C_1 and R_2.

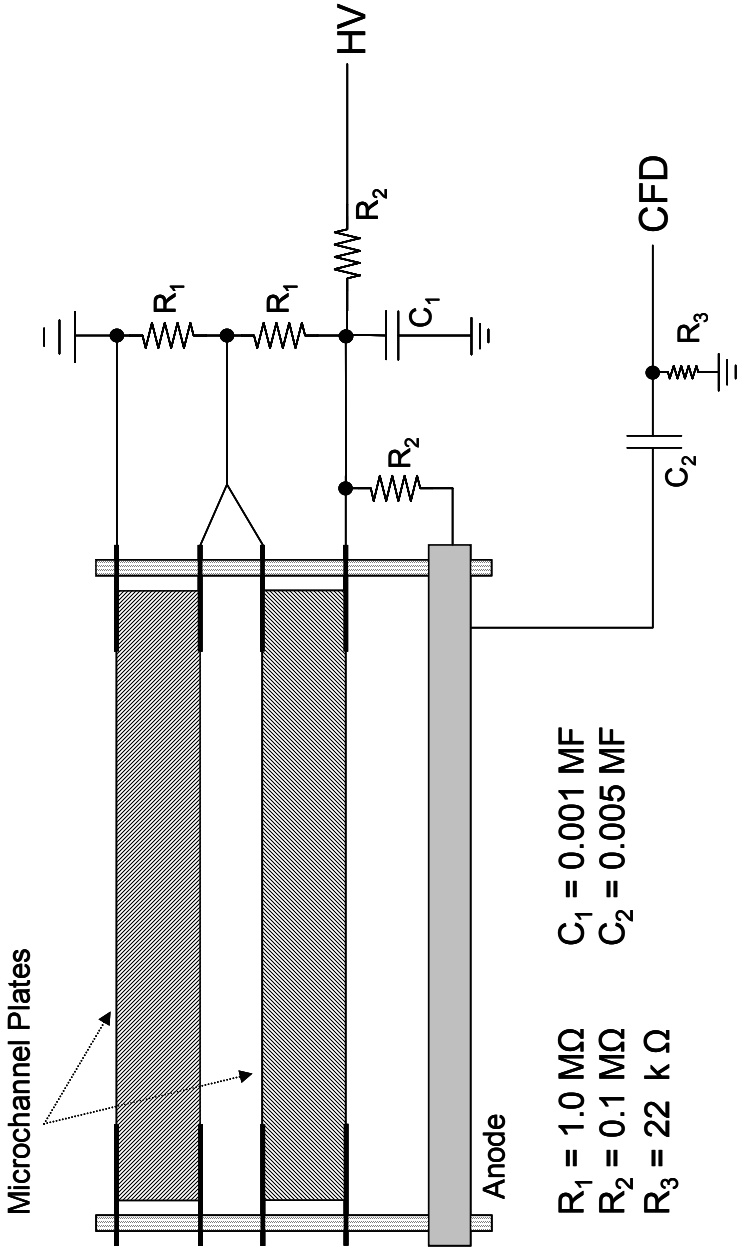


Fig. 3-8: Schematic of micro-channel plate detector assembly for start and stop detectors.

This filter attenuates the 60 Hz noise from the 120 power supplying the HV power supply. The signal is picked off the anode through a low pass filter, C_2 and R_3.

It is periodically necessary to optimize the MCP detector array. This is due to the inevitable loss in gain from the degradation of the electrode material over time and/or, experiments in which a particular mass is to be detected (detector response depends on the velocity of the charged particle [38]). The procedures for optimizing the start and stop detectors differ.

The signal amplitude from the start detector varies with the mass of each charged particle that is detected. If the amplitude is above the discriminator threshold (described later in this chapter) it will be recognized as a legitimate start. The start signal should reflect the impact of a primary ion on the target. False starts contribute to noise in the secondary ion spectrum by recording uncorrelated secondary ions. To minimize this type of noise, the yield, i.e. number of secondary ions per primary ion, can be plotted as a function of the discriminator threshold. Figure 3-9 is a plot of the yield of the phenylalanine quasi-molecular ion $[M-H]^-$ from 22 keV Au_3^+ bombardment as a function of this threshold. As the threshold is increased, the number of false starts decreases, resulting in a concomitant increase in the yield. The yield increases until the number of false starts is effectively eliminated, resulting in a plateau in this curve which reflects the true experimental secondary ion yield for phenylalanine. A further increase in the threshold will result in a decrease in the duty cycle of the instrument by decreasing the true start rate. The detector is then optimized by setting the threshold at a point where the yield plateaus giving a minimum of false starts and a maximum start rate.

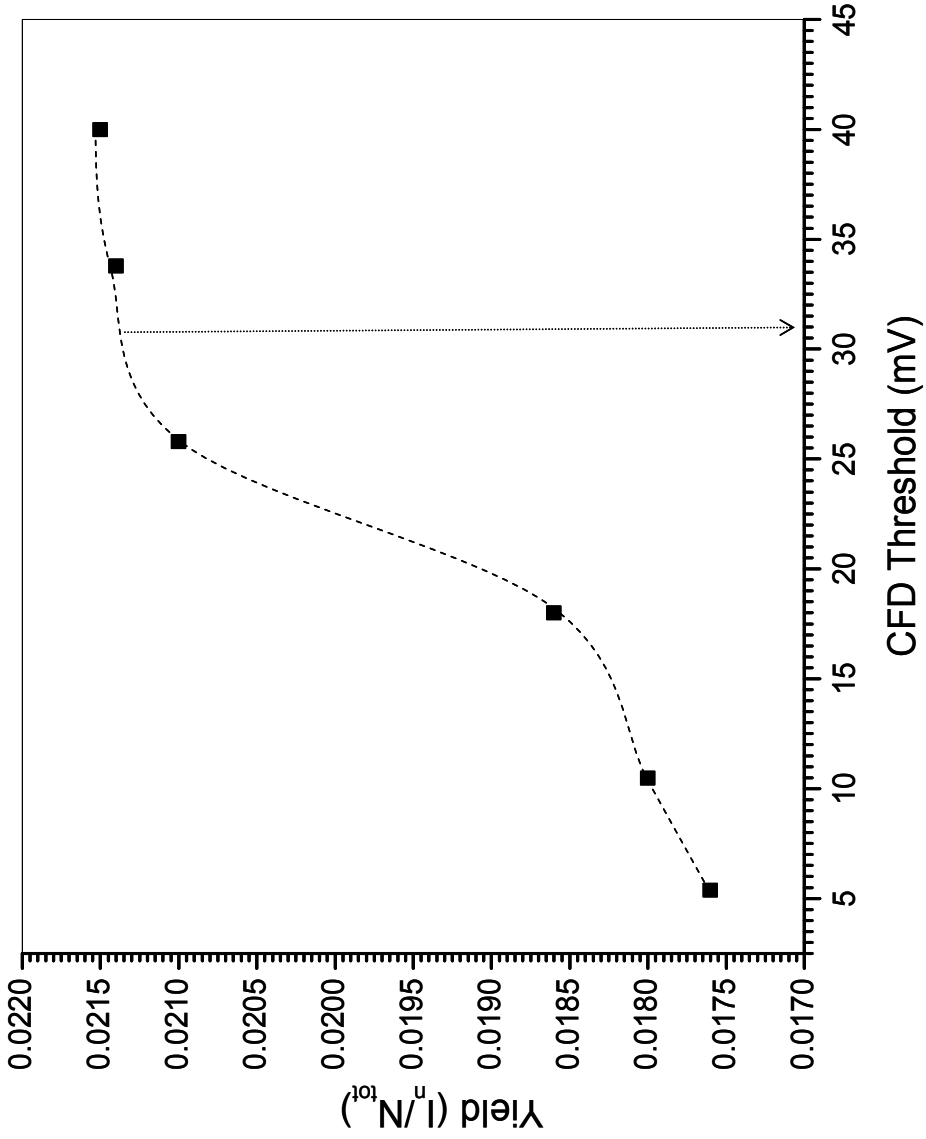


Fig. 3-9: Yield of phenylalanine, $[M\text{-}H]^-$ as a function of CFD threshold for start detector. Bias and CFD threshold on stop detector are constant.

Optimizing the stop detector requires a different procedure. In this case the discriminator threshold is kept at a minimum. This insures detection of all ions. The detector bias can be increased which results in an increase in gain. The problem is that, the potential is increased, the noise generated by the MCP itself also increases [38]. This degrades the signal to noise, S/N, ratio for the measurement. Figure 3-10 is a plot (at constant stop discriminator threshold) of the S/N ratio for PO_3^- from 22 keV Au_3^+ bombardment of a zirconium phosphate target. The S/N ratio increases until a maximum is reached. A further increase in detector bias results only in an increase in noise. For optimum detector performance one would choose a bias that gives a maximum S/N ratio.

Timing Electronics

The analog pulse from the MCP is transferred to a Constant Fraction Discriminator, CFD, (935 Quad 200 MHz CFD, Ortec) where it converts the analog pulse to a logic pulse (NIM). Triggering is accomplished at some fraction of the amplitude of the incoming pulse. Signal walk, i.e. variations in trigger time due to variations in incoming signal amplitude, can result in a decrease in time resolution [39]. The CFD reduces this type of peak broadening by splitting the incoming unipolar signal into two portions. One signal is inverted and attenuated by 20%. The two are then recombined into a bipolar signal. The output is triggered at the point where the signal polarity changes from negative to positive (zero-crossing). This gives a logic pulse independent of the amplitude of the incoming signal. The discriminator threshold, i.e. the minimum amplitude of the incoming pulse which will trigger the CFD, can be

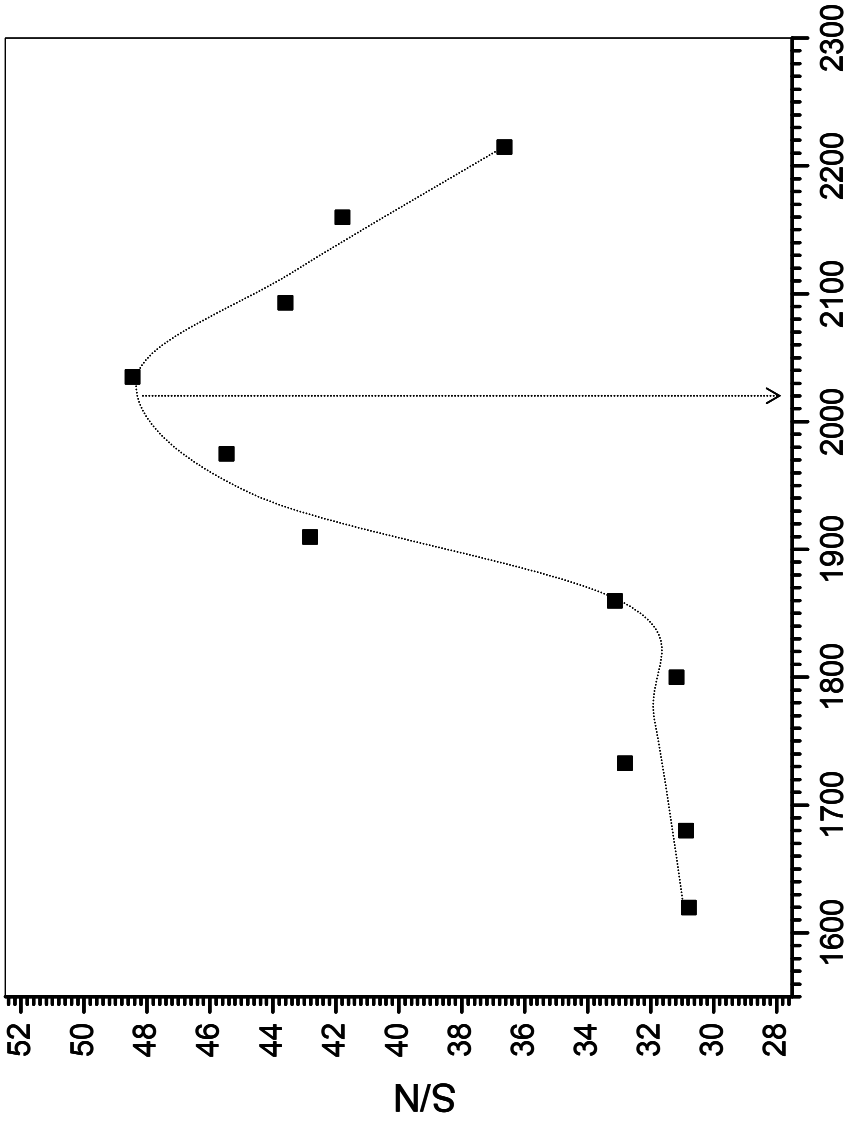


Fig. 3-10: Signal to noise ratio for PO_3^- as a function of stop detector bias.

adjusted from 0 – 1000 mV. The width of the output pulse (blocking width) is adjustable <5 ns - >1 μs. If two ions are separated by a time less than the CFD blocking width, the second ion will go undetected. Setting the blocking width to a minimum for the stop detector can compensate this for problem. Pile-up, in the Time-to-Digital Converter (described in the following paragraph and Chapter IV), can occur when more than one start signal from the CFD is transmitted during the same analysis period. The blocking width for the start detector should be set at the maximum, not to exceed the ToF analysis duration, to reduce pile-up.

The logic pulse is passed from the CFD to the Time-to-Digital Converter, TDC. The TDC is described in greater detain in Chapter VI. The TDC then transfers the data to a personal computer for storage and subsequent analysis.

Figure 3-11 illustrates signal processing for both the primary and secondary ion time of flight legs.

1.) Signal from the pulse plate is routed through the CFD to a separate TDC (Schmidt Ind., Houston, TX) labeled TDC 1, and serves as a start signal for the primary ion ToF.

2.) The primary ion strikes the target which in turn ejects electrons. These electrons are steered toward the primary ion MCP detector array generating a signal which is routed through the CFD. This signal serves two purposes:

a.) The output from the CFD is then routed to TDC 1. It is the stop signal for the primary ion ToF. This allows for verification of the identity of the primary ion.

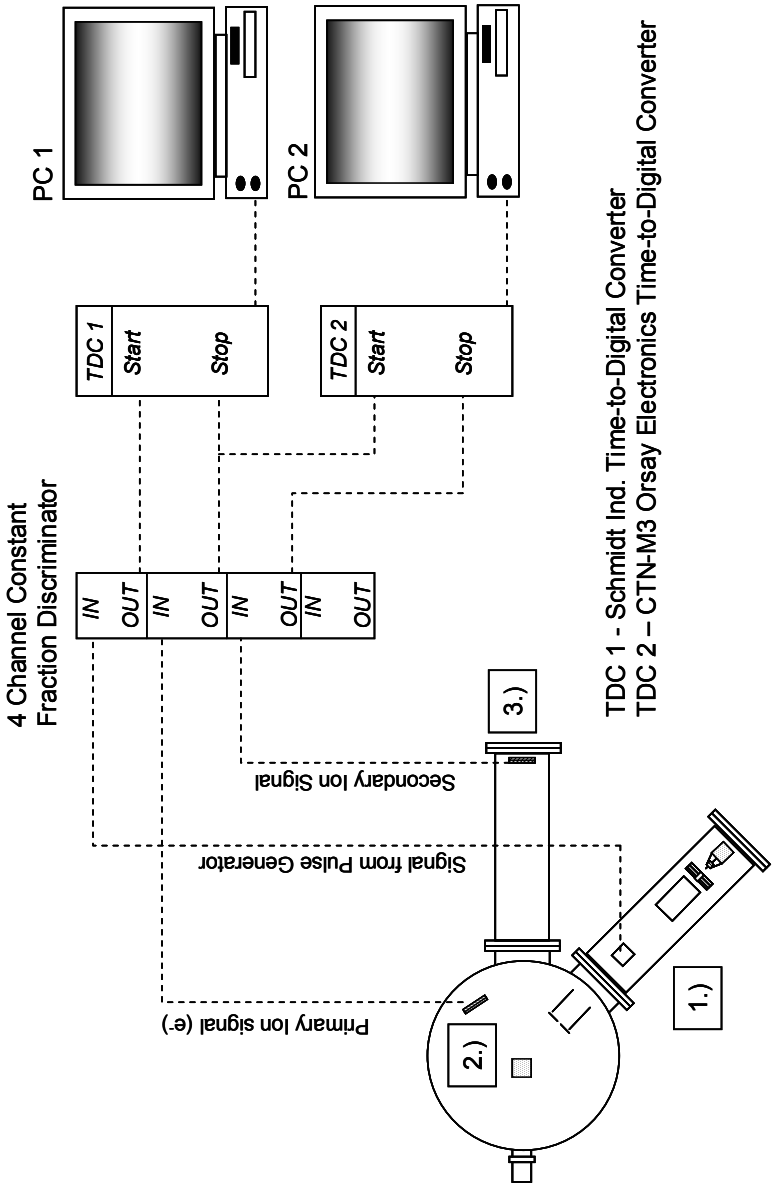


Fig. 3-11: Signal processing for SYS VII.

b.) The output is also routed to the CTN-M3, labeled TDC 2. This serves as a start signal for the ToF mass analysis of any secondary ions ejected by that particular primary ion.

3.) The secondary ions arrive at the secondary ion MCP array generating a signal that is processed through the CFD to TDC 2. This allows for ToF mass analysis of the secondary ions.

CHAPTER IV

DATA ACQUISITION

Data Acquisition and Analysis in the Event-by-Event Bombardment / Detection Mode

The data presented here were collected using Time-of-Flight Secondary Ion Mass Spectrometry, ToF-SIMS, with the event-by-event detection of secondary ion(s) ejected by the impact of a single primary ion. An event, in this case, is defined as the interaction of a single, energetic projectile with a target surface. As a result of this interaction a number of secondary particles may be ejected from the surface to include secondary electrons, secondary ions, and neutrals (Figure 4-1). If a potential is applied to the target; secondary particles of the same charge will be accelerated towards a detector where they can be identified. In principle, one can create a mass spectrum resulting from a single projectile impact. However, on average the number of secondary ions detected from a single event is very small, e.g. the yield for phenylalanine is ~0.02 ions/projectile. Typically a few hundred thousand to several million single events are collected, yielding a mass spectrum statistically representative of the target.

Provided the event is the result of a single projectile impact, any secondary ions that are ejected will be from a localized area, i.e. the area perturbed by a single projectile. This area can be as small as 10 nm in diameter [13] implying that any co-emitted ions will be spatially and/or chemically related on this scale. Detection of these co-emitted, or coincidental ions is useful for probing surface structure [15], investigating

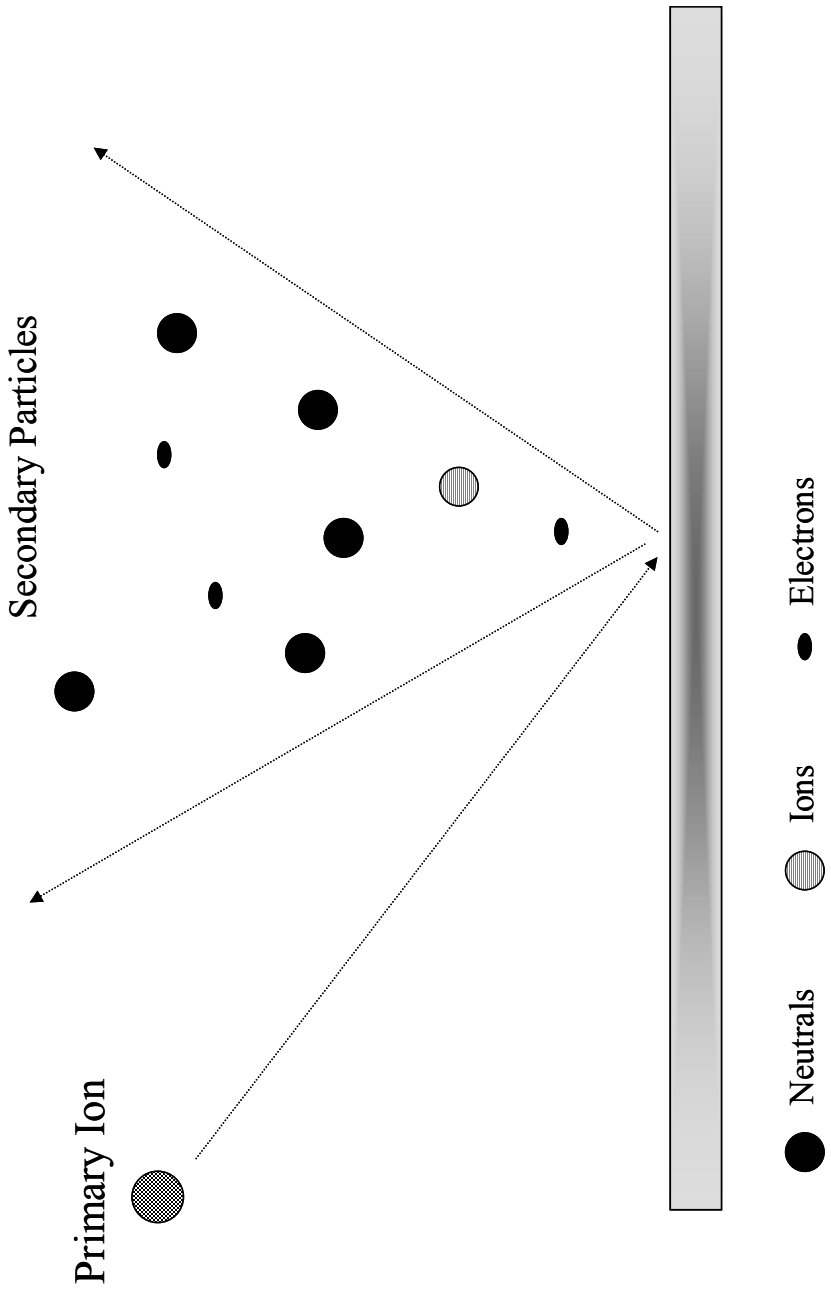


Fig. 4-1: Interaction of an energetic projectile with a surface.

chemical micro-homogeneity [36], for nano-particle analysis [12], and for investigating the fundamentals of ion/surface interactions [40, 41]. Analysis in the event-by-event mode requires hardware capable of recognizing individual events and software for interpreting the raw data.

Hardware

In the event-by-event approach, single ions are detected by registering pulses from a detector. Time-to-Digital Converters, TDC's, are particularly well suited for these types of measurements [42] and were used exclusively for the collection of the experimental data presented in following chapters.

The data collection process is triggered when a signal from the start detector, at time t_0, is passed to the TDC. As secondary ions strike the stop detector, the pulses are processed in the TDC and their arrival times, t_a, are recorded. This information is digitized and passed to a PC for user analysis. The secondary ions can be identified by their flight time, t_f, which is $t_a - t_0$. Two TDC's are used in our laboratory, the CTN-M3 and –M4.

CTN-M3

The CTN-M3 is the third generation in the CTN series. This TDC was developed and manufactured at the Institute de Physique Nucleaire, Orsay, France. Several improvements were made in this version compared to previous ones however, only the data format is addressed here.

There are a number of data format options on the -M3, of these program 6 and 8 are used. Program 6 provides the flight times for the secondary ions detected, with a maximum resolution of 0.4 ns. Both resolution and analysis time can be set by the operator when using program 6. In this mode it is not possible to assign secondary ions to individual events. Consequently information about co-located or coincidental ions is lost. If this information is not needed, program 6 is preferred since RAM and CPU requirements are minimized.

Program 8 provides information that allows the user to fully exploit the event-by event mode of analysis. This program provides flight times for each secondary ion, which detector registered their arrival (up to 8 independent detectors can be used), and an end-of-event flag that indicates the event is over. The data are encoded in two 16 bit words (Fig. 4-2). The most significant bit of each word is reserved and is either a one for the first word or a zero for the second word. Bits 14 – 10 of the first word indicate which detector the ion was registered on, or if no ions were detected, contains the end-of-event flag. The remaining 10 bits of the first word and the first 15 bits of the second word give the arrival time of the ion. Each event consists of $2n+2$ words where n is the number of secondary ions detected in the event. The resolution (0.4 ns/ch) is not user configurable in program 8 so the analysis time sets the number of channels each mass spectrum will have.

First Word

1	L_1	L_2	L_3	L_4	L_5	2^{24}	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	2^{15}

Label — Time

Second Word

0	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

Time

Labels

	L_1	L_2	L_3	L_4	L_5
Detector 1	0	0	0	0	0
Detector 2	0	0	0	0	1
Detector 3	0	0	0	1	0
Detector 4	0	0	0	1	1
Detector 5	0	0	1	0	0
Detector 6	0	0	1	0	1
Detector 7	0	0	1	1	0
Detector 8	0	0	1	1	1
End-of-Event	1	0	0	1	0

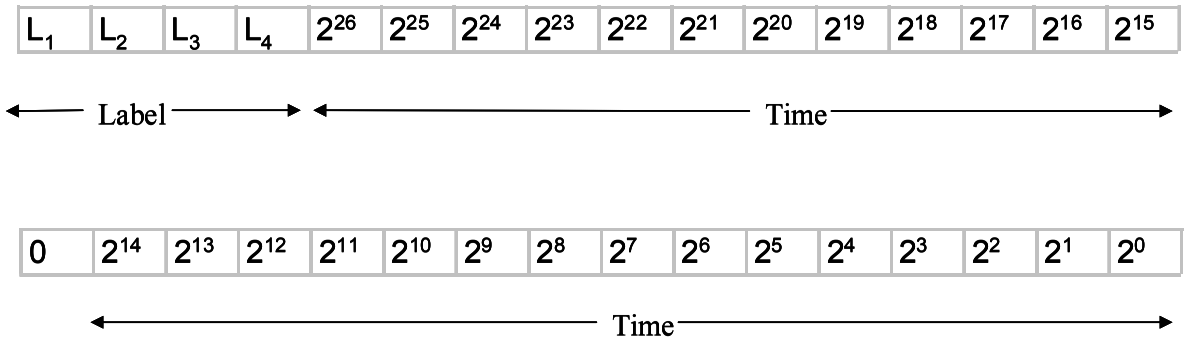
Fig. 4-2: Data format for program 8, two 16 bit words (CTN-M3).

CTN-M4

The -M4 is similar to the -M3 with three exceptions: 1) Data acquisition on the -M3 is operator-initiated by manually resetting the on-board memory and toggling the start switch. This procedure is handled with software in the -M4. 2) The 25 pin DIN to 68 pin SCSI II conversion box is not needed. The -M4 has a 68 pin SCSI II output which is compatible with the DAQ card. And, 3) Data labels for detector numbers and the end of event flags are different (Fig. 4-3). Like the -M3, several data format options are available. At present, only program A, which provides the same information as does program 8 on the -M3, is used.

Data Acquisition Interface Card

Interface of the TDC to the PC is via a National Instruments PCI-DIO 32HS (777314-01) data acquisition card. The card consists of 32 digital I/O lines with associated control lines for organizing the handshaking transfer of data between the TDC and PC. The card is inserted in any free PCI slot in the PC. A program from National Instruments, "Measurement and Automation", must be executed after installation of the card. This software initializes the card and assigns it a number. *This must be card number 1 for the acquisition software to function correctly.* Connection of the card to the TDC is done either directly, (using a 68 pin SCSI II cable) or through a custom interface box (for converting 25 pin DIN to 68 pin SCSI II, Appendix C), depending on which TDC is used.



Labels

	L_1	L_2	L_3	L_4
Detector 1	0	0	0	0
Detector 2	0	0	0	1
Detector 3	0	0	1	0
Detector 4	0	0	1	1
Detector 5	0	1	0	0
Detector 6	0	1	0	1
Detector 7	0	1	1	0
Detector 8	0	1	1	1
End-of-Event	1	0	0	0

Fig. 4-3: Data format for Program A, two 16 bit words (CTN-M4).

Software

The Total Matrix of Events, TME©, software is used for acquiring and processing data collected in event-by-event mode. Although developed specifically for the -M3 and –M4, minor modifications to the source code would enable its use with most commercially available TDC's.

The raw data is acquired and transferred to file in the PC. This is done for events one through m_{total}, where m_{total} is the total number of events acquired (Fig. 4-4). Initially the raw data is categorized by the number of secondary ions detected per event and the detector which recorded their arrival. For instance all events, in which one secondary ion was detected (1-ion event), are placed in a matrix with 2 columns and m_1 rows where m_1 is the total number of 1-ion events. The first column contains the arrival time of the secondary ion and the second column contains the detector information. Events in which two secondary ion were detected (2-ion events) are stored in a 4 x m_2 matrix (in this case m_2 is the number of 2-ion events). As with the 1-ion event, each row defines an event. The first and third columns contains the arrival time of each secondary ion. The second and fourth columns contain the detectors that registered their arrival. Thus for an n-ion event, where n is the number of secondary ions detected and m_n is the total number of n-ion events, a $2n$ x m_n matrix is required. Collectively these matrices are the TME©. At present the TME© can store events in which up to 50 different secondary ions are detected. However, the TDC can register > 2000 secondary ions from a single start. Some modification to the source code would make it possible to collect and store these types of events.

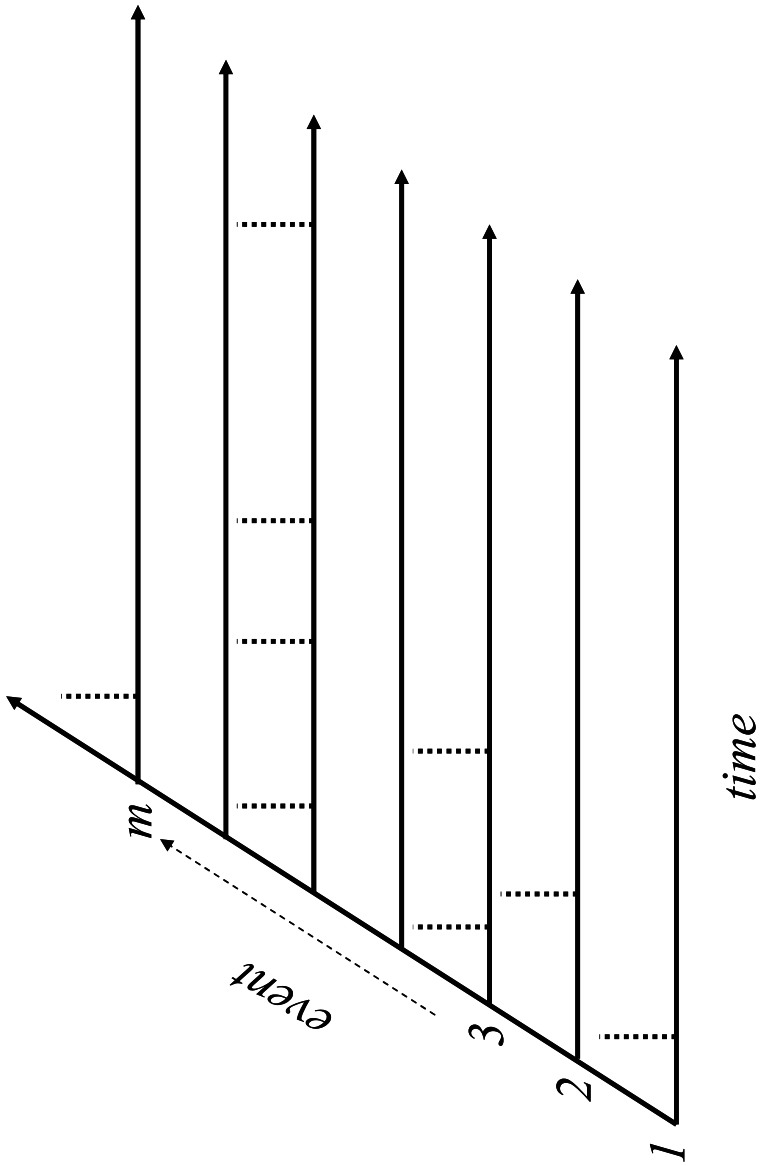


Fig. 4-4: Acquisition and storage of individual events.

A mass spectrum, from the raw data, is created by defining a region of interest, ROI, using the TME©. Any ion having an arrival time falling within the ROI is included in the mass spectrum. A total mass spectrum (Fig. 4-5) is created by setting the ROI minimum at zero and the ROI maximum ch_{max}. The numerical value of ch_{max} can be calculated from knowledge of the analysis duration and time resolution. For instance if the time resolution is 0.4 ns per channel and the analysis duration is 13 μs then:

$$ch_{\max} = \left(\frac{13\ \mu\text{s}}{0.4\ \frac{\text{ns}}{\text{channel}}} \right) \left(1000 \frac{\text{ns}}{\mu\text{s}} \right) = 32,500 \text{ channels} \qquad \textbf{Eq 4-1}$$

Within the total mass spectrum the user can then search for specific types of events. One example would be to set ROI around a peak of interest, a window ion, and search individual matrices or a combination of several matrices for events in which that ion was detected. The TME© will then search and display the results, in the form of a histogram, from all events in which the window ion was detected. For instance, Figure 4-6 is a spectrum that consists of all secondary ions that were detected in coincidence with, or from the same events, in which the window ion (m/z=164, phenylalanine [M-H]) was detected. Additional constraints can be placed on how the spectra are generated. Figure 4-7 is similar to the previous spectrum, however this search is restricted to only the 2-ion matrix. It is also possible to define multiple ROI's as part of the search criteria. Figure 4-8 is a mass spectrum created by setting a window on m/z=172 (deuterated phenylalanine [M-H]-) from Fig. 4-6. This spectrum includes all secondary ions that were detected in coincidence with *both* the non-deuterated *and* the

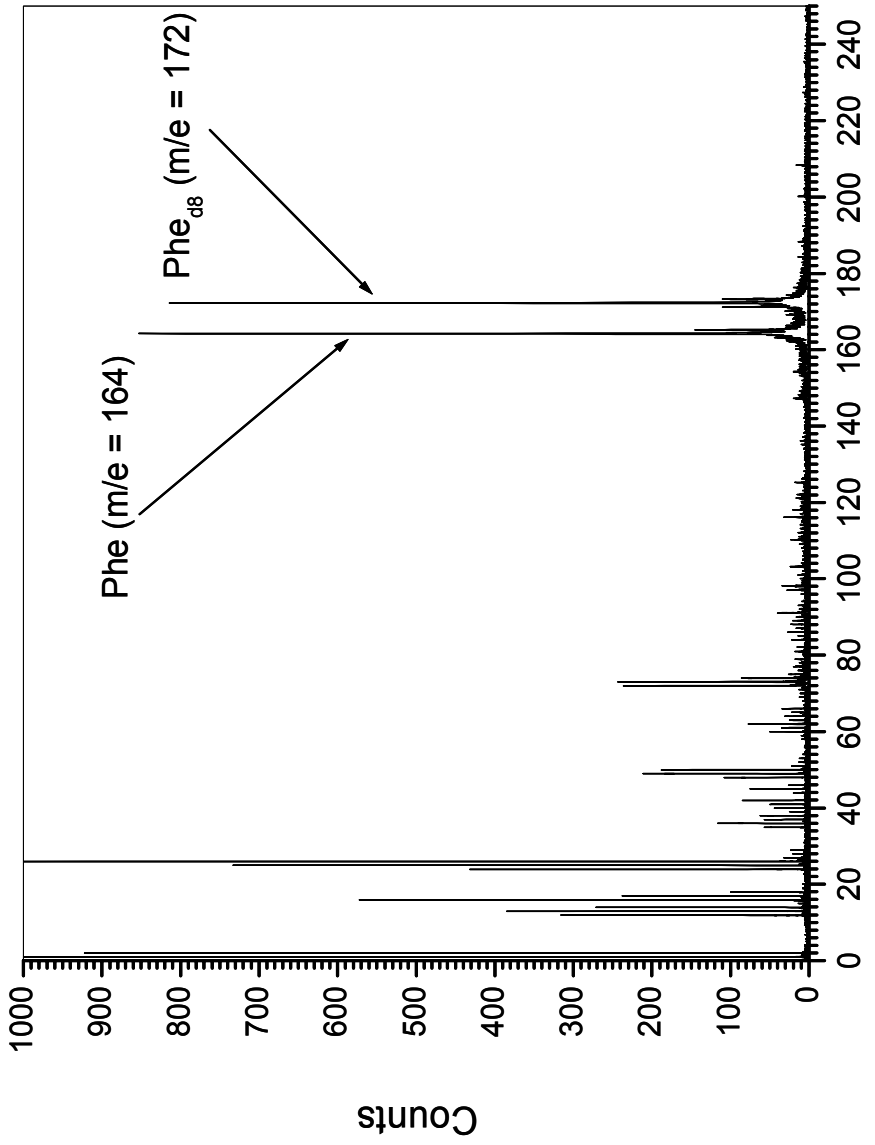


Fig. 4-5: Total negative ion mass spectrum from 27 keV Au_3^+ bombardment of a mixed deuterated/non-deuterated phenylalanine target.

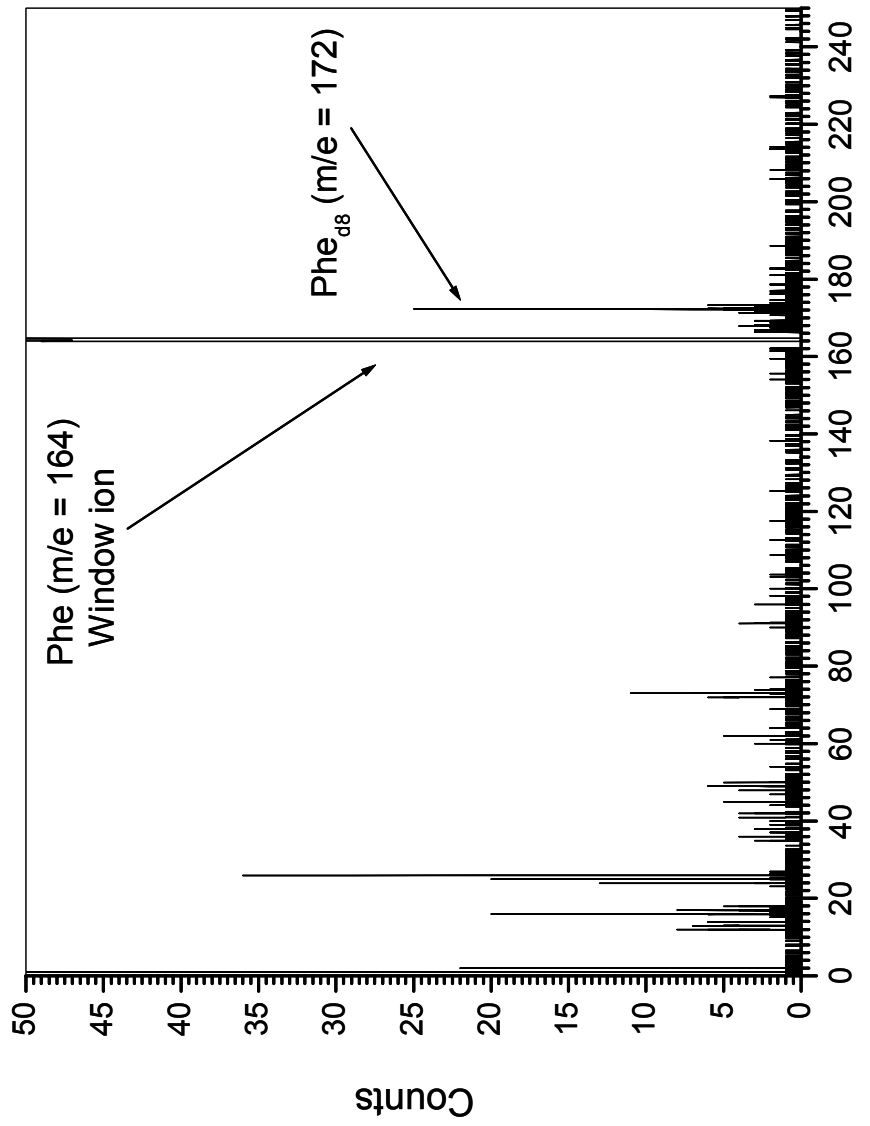


Fig. 4-6: Coincidence w/ m/e = 164 negative ion mass spectrum from 27 keV Au_3^+ bombardment of a mixed deuterated/non-deuterated phenylalanine target.

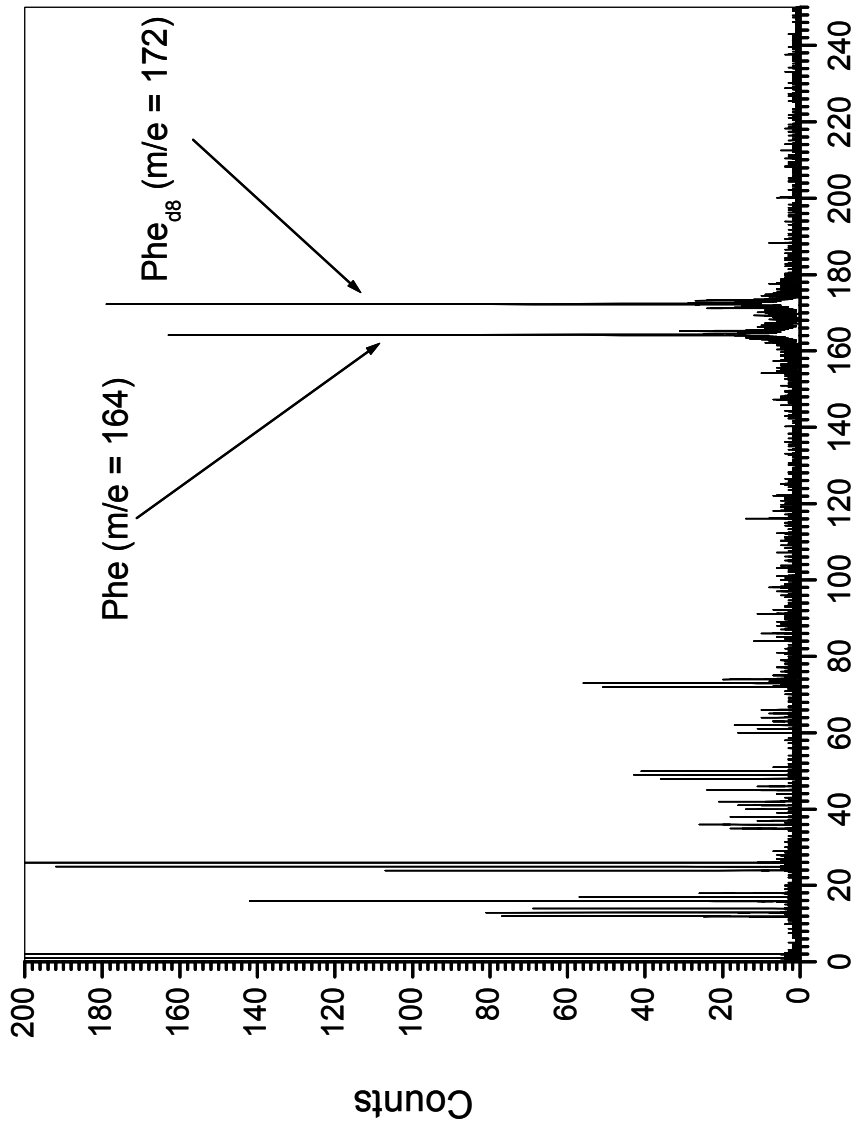


Fig. 4-7: Total negative ion mass spectrum of two-ion emission events from 27 keV Au_3^+ bombardment of a mixed deuterated/non-deuterated phenylalanine target.

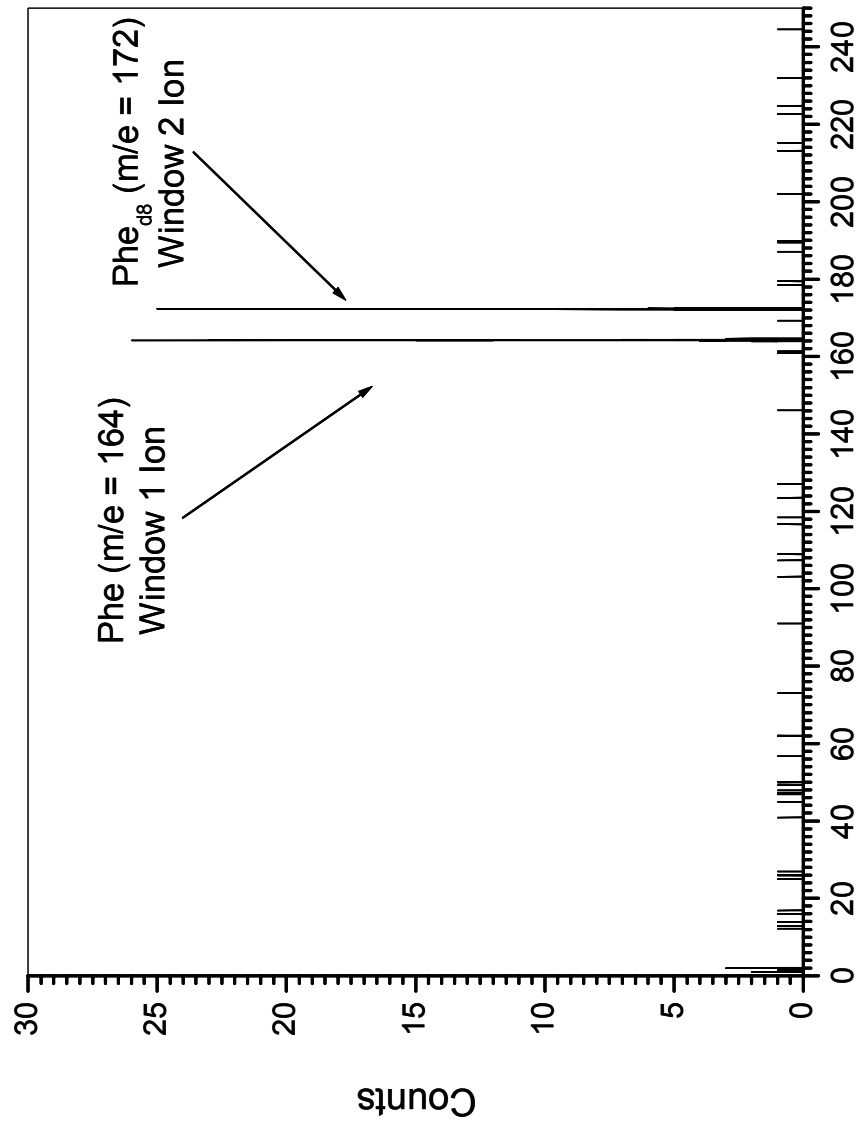


Fig. 4-8: "Double" coincidence negative ion mass spectrum from 27 keV Au_3^+ bombardment of a mixed deuterated/non-deuterated phenylalanine target.

secondary ions that were detected in coincidence with *both* the non-deuterated *and* the deuterated phenylalanine quasi-molecular ions.

CHAPTER V

MULTIPLE SECONDARY ION EMISSION EVENTS FROM keV ATOMIC AND POLYATOMIC PROJECTILE BOMBARDMENT

As mentioned in Chapter I, the efficiency of a keV projectile to induce emission of secondary ions from a target varies with the energy, the number of constituent atoms in the projectile, their mass, and the nature of the target. The yields of the phenylalanine quasi-molecular ion, $[M-H]^-$, divided by the number of projectile constituents are shown as a function of projectile energy per atom in Figure 5-1. The trends depicted here have been described elsewhere [30, 31, 43, 44]. Secondary ion yields increase in a supra-linear fashion as the complexity and velocity of the projectile increases. A figure of merit to quantify this degree of supra-linearity is the enhancement factor (Eq. 2-1). An enhancement factor, $\varepsilon = 5.4$, has been calculated from the yields of $[M-H]^-$ from Au_3^+ and Au^+ at 15 keV/atom. These yields reflect contributions from events in which both single and multiple secondary ions are detected.

In this bombardment regime, one might expect polyatomic projectiles to be more efficient at generating events in which multiple secondary ions are ejected than equal velocity atomic ions. This assumption is based on molecular dynamics simulations [45] and the experimental observation of enhanced secondary ion yields from polyatomic projectile impacts [31]. An initial study confirms the expected trend for coincidental ion yields vs. projectile characteristics [43]. However, events in which two or more secondary ions are detected are rare. Figure 5-2 is a secondary ion multiplicity report for

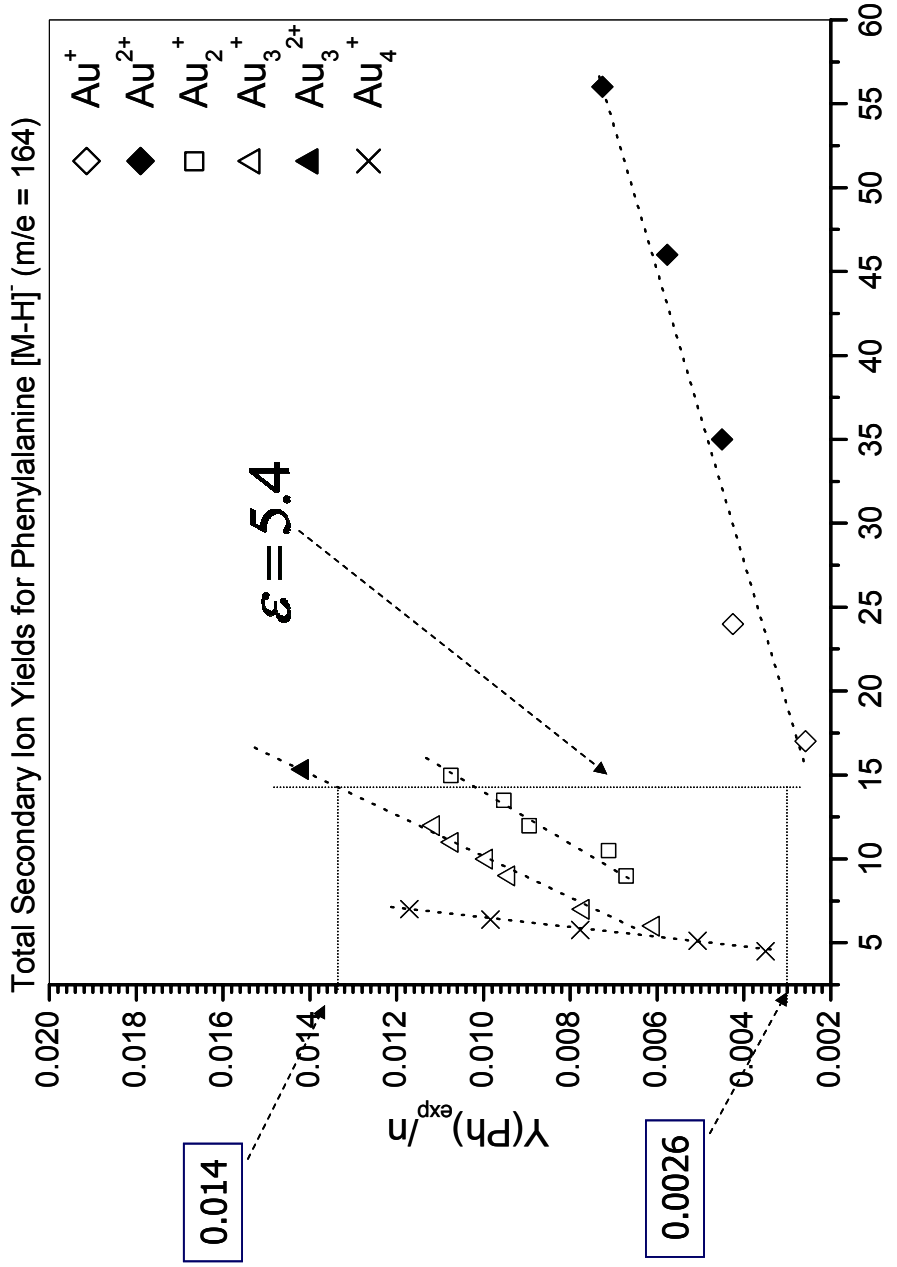


Fig. 5-1: Total secondary ion yields for the Ph molecular ion $(M-H)^-$ on a per atom basis as a function of the energy per atom of the Au_n^{m+} projectiles, n = 1 to 4, m=1,2. Lines are a guide for the eye.

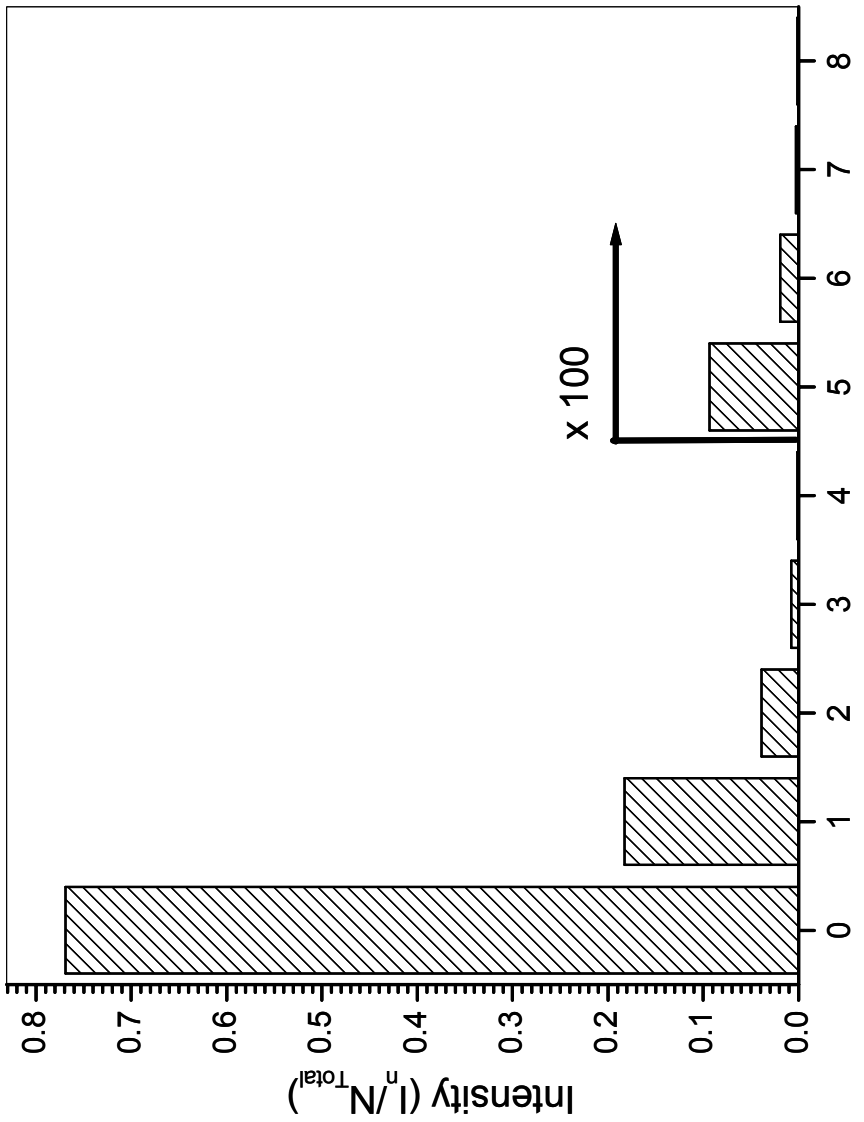


Fig. 5-2: Secondary ion multiplicity report for 23 keV Au_4^+ bombardment of a phenylalanine target.

23 keV Au4+ bombardment of a phenylalanine target. Ion multiplicity is defined here as the number of *different* secondary ions detected per event. On the x-axis is a number that corresponds to the total number of secondary ions detected in a single event. The y-axis is the intensity of these events expressed as a percentage of all recorded events. In this example most events (~74%) are null events, i.e. events in which no secondary ion was detected. Folded into these non-productive events are a few productive ones which yield one or more secondary ions. One now wonders why a gold atom, as part of a cluster, is more effective that a single gold atom for the emission of this ion. Is there a disproportionate contribution, to these yields, form events in which multiple secondary ions are detected? To answer the latter question we will examine yields, from single and multi-ion emission events as a function of projectile characteristics.

Gold atomic and polyatomic ions ($Au_1^+ - Au_4^+$) were used with impact energies in the 17-56 keV range. The target potential was kept constant while the energy of the primary ion was varied. The collection geometry for secondary ions may vary from one point on the target to the next due to possible inhomogeneities in the electric field in the secondary ion acceleration region. Keeping a constant target potential insures that the transmission and detection efficiency for the secondary ions remained relatively constant. The impact angle, ϕ, of the primary ion relative to the target normal depends on the primary ion energy and the target bias. This relationship is expressed formally in the following equation:

$$\phi = \cos^{-1}\left(\frac{U_1 \cos^2\Theta - U_2}{U_1 - U_2}\right)^{\frac{1}{2}} \qquad \textbf{Eq. 5-1}$$

where U_1 is the kinetic energy of the primary ion, U_2 is the target bias (constant), and Θ is the angle of the primary ion axis relative to the target normal (60^o). For these experiments $36^o < \phi < 48^o$. The influence of this small variation on the secondary ion yields is negligible [44]. The secondary ion yields, Y_{SI}, are defined by the following equation:

$$Y_{SI} = \frac{I_{SI}}{N_{tot}} \qquad \textbf{Eq. 5-2}$$

where I_{SI} is the number of counts in the secondary ion peak of interest and N is the total number of recorded events. Secondary ion yields for various projectiles having the same velocity can be compared with the enhancement factor (Eq. 2-1).

Events yielding both single and multiple secondary ions were identified from the record of all events with the TME© software. The latter allowed for the data to be categorized according to the number of secondary ions detected per event. Thus it is possible to plot the yields of secondary ions from single ion detection events, from events where two secondary ions were detected, and so on.

Figure 5-3 is a plot of the yields of the phenylalanine $[M-H]^-$ as a function of projectile characteristics from events in which only one secondary ion is detected. There is little if any enhancement between the different polyatomic projectiles when compared to overall secondary ion yields (Fig. 5-2). Note that at 50 keV/atom, the atomic projectile offers similar yields. Looking now, at cases where two secondary ions are detected, Fig. 5-4, the supra-linear relationship between yields and projectile characteristics is clearly visible and the degree of supra-linearity increases as the number

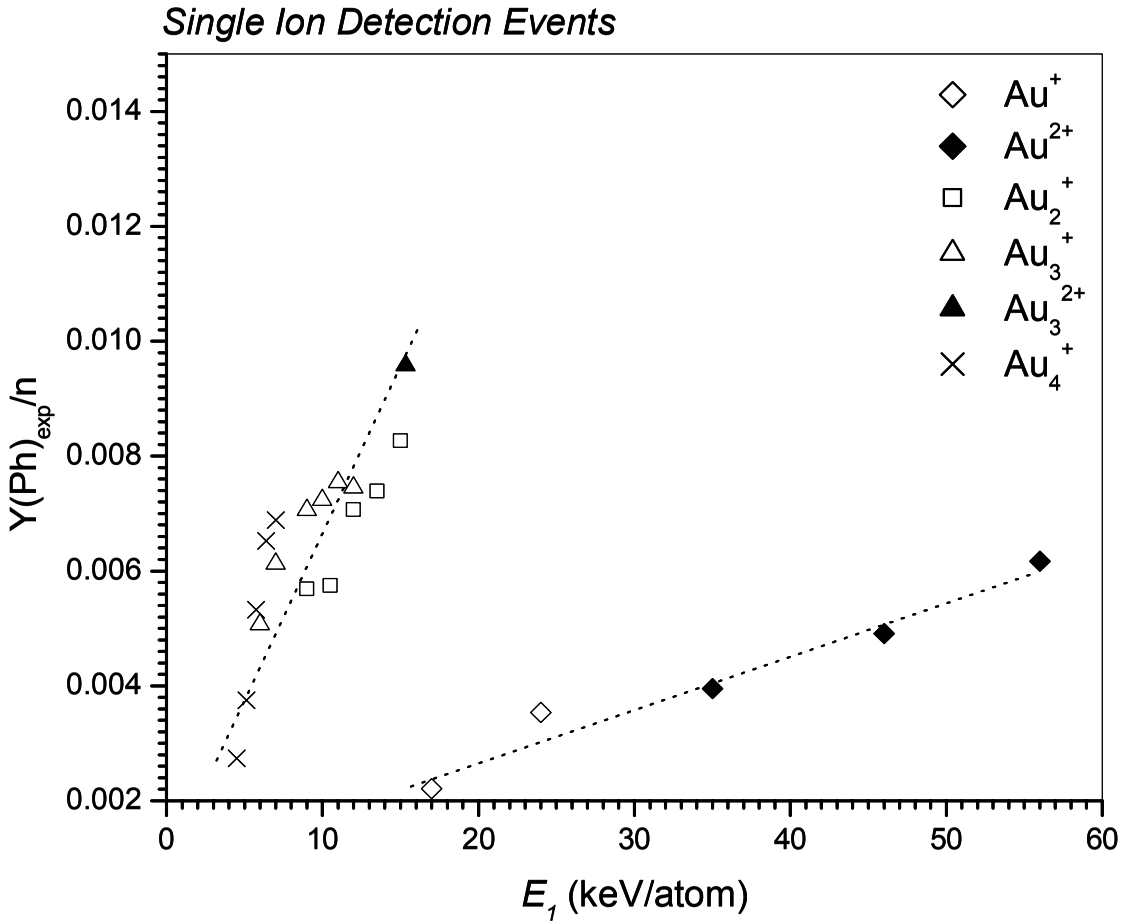


Fig. 5-3: Secondary ion yields for the Ph molecular ion $(M-H)^-$ on a per atom basis as a function of the energy per atom of the Au_n^{m+} projectiles, n = 1 to 4, m=1,2 from one-ion detection events. Lines are a guide for the eye.

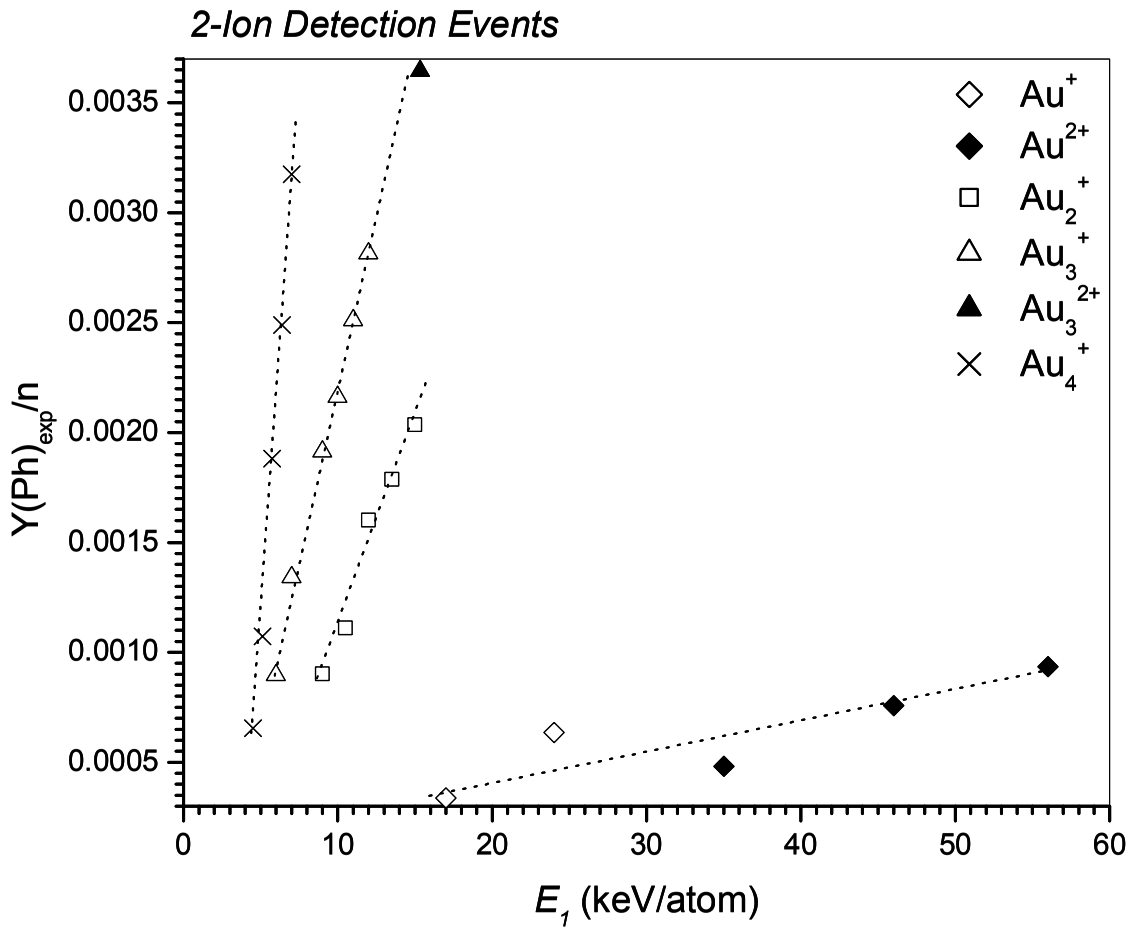


Fig. 5-4: Secondary ion yields for the Ph molecular ion $(M-H)^-$ on a per atom basis as a function of the energy per atom of the Au_n^{m+} projectiles, n = 1 to 4, m=1,2 from two-ion detection events. Lines are a guide for the eye.

of secondary ions detected increases (Fig 5-5 and 5-6). An enhancement factor for the yield enhancement for Au3+ versus Au+, at 15 keV/atom, has been plotted for single ion, 2-ion, 3-ion, and 4-ion emission events in Fig. 5-7. A non-linear relationship exists between the enhancement factor and the number of molecular secondary ions detected. The yields of atomic secondary ions will now be examined using this same approach.

The secondary ion yields for the deuterium ion, d^-, from deuterated phenylalanine, are shown in Fig. 5-8. It has been reported that polyatomic projectiles provide little or no enhancement in the secondary ion yields of hydrogen [31] and our data are in agreement with previous observations. Figure 5-9 – 5-12 shows the d^- yields as a function of projectile characteristics and the number of secondary ions detected per impact. Clearly there is an enhancement in yields where multiple secondary ions are detected (Figs. 5-10 and 5-11). This enhancement is not evident in the overall secondary ion yields of d^- (Fig. 5-8). The reason is that the yields in these multi-ion emission events are insignificant, e.g. $\sim 10^{-5}$ in the 4-ion events versus 10^{-3} for the overall yields.

The yield data from single and multiple ion detection events suggests that the collision cascades can be broadly categorized into two groups, linear and non-linear. Both types of cascades contribute to the overall secondary ion yield with relative contributions dictated by projectile characteristics. Linear collision cascades result mostly in the ejection of single secondary ions and only the energy and mass of the primary ion affects the yield. This case confirms what would be expected from this type of collision cascade [19]. The contribution of the non-linear collision cascades dominates the overall secondary ion yields for the phenylalanine molecular ion, Fig. 5-1,

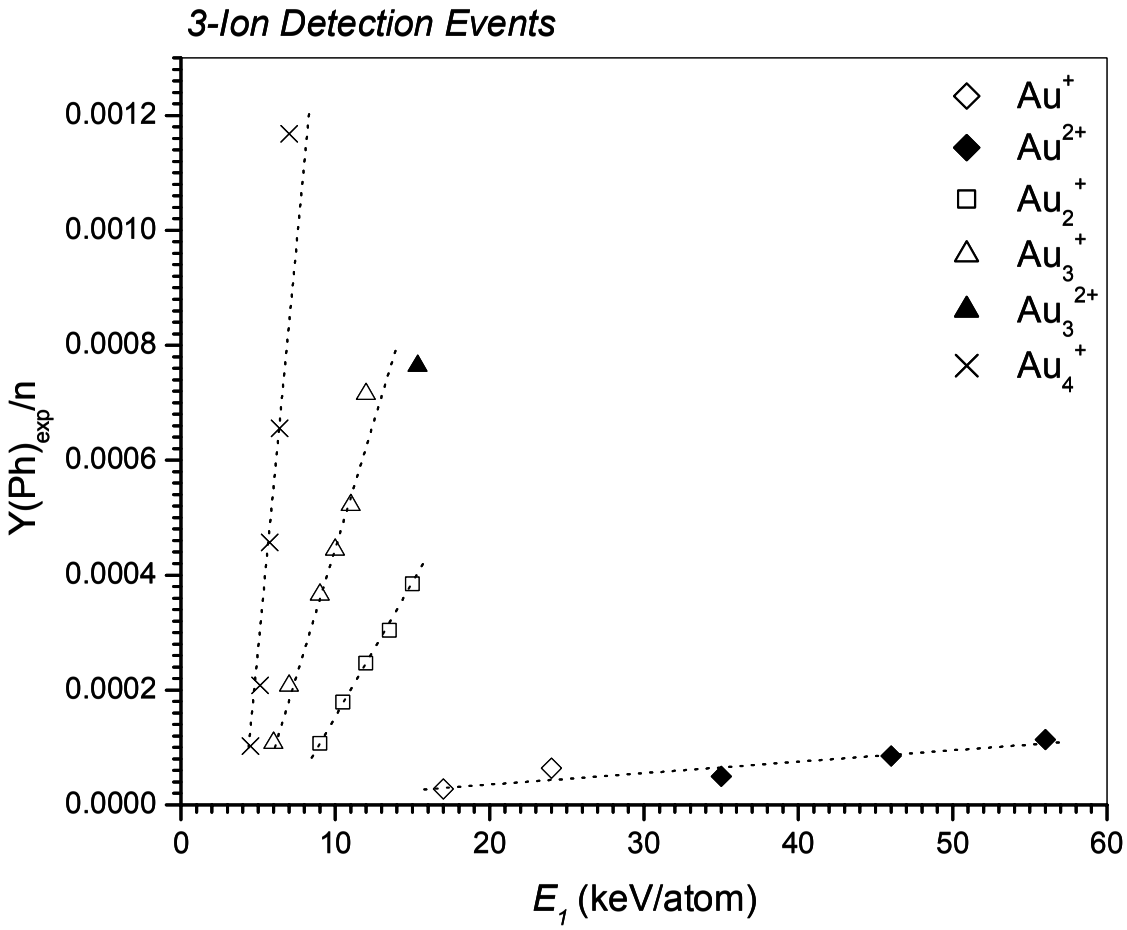


Fig. 5-5 Secondary ion yields for the Ph molecular ion $(M-H)^-$ on a per atom basis as a function of the energy per atom of the Au_n^{m+} projectiles, n = 1 to 4, m=1,2 from three-ion detection events. Lines are a guide for the eye.

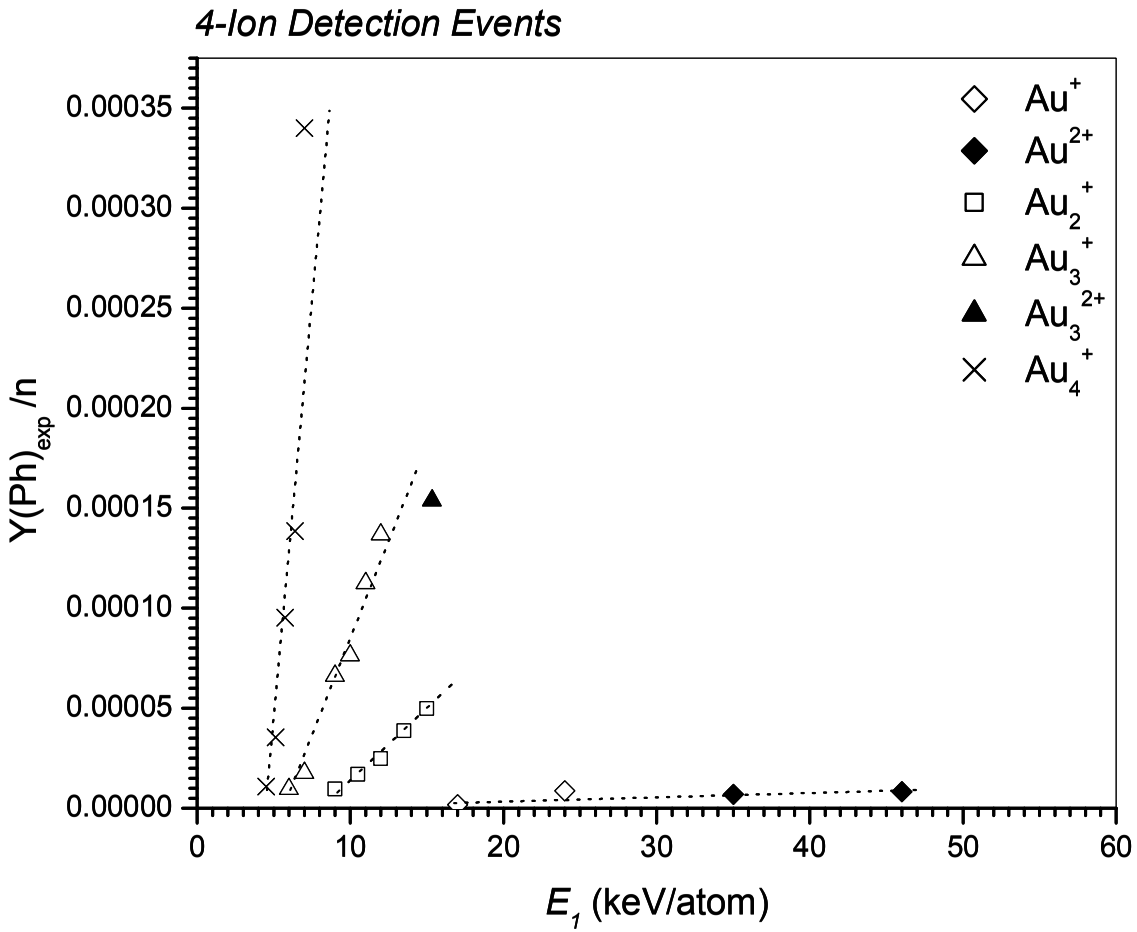


Fig. 5-6: Secondary ion yields for the Ph molecular ion $(M-H)^-$ on a per atom basis as a function of the energy per atom of the Au_n^{m+} projectiles, n = 1 to 4, m=1,2 from four-ion detection events. Lines are a guide for the eye.

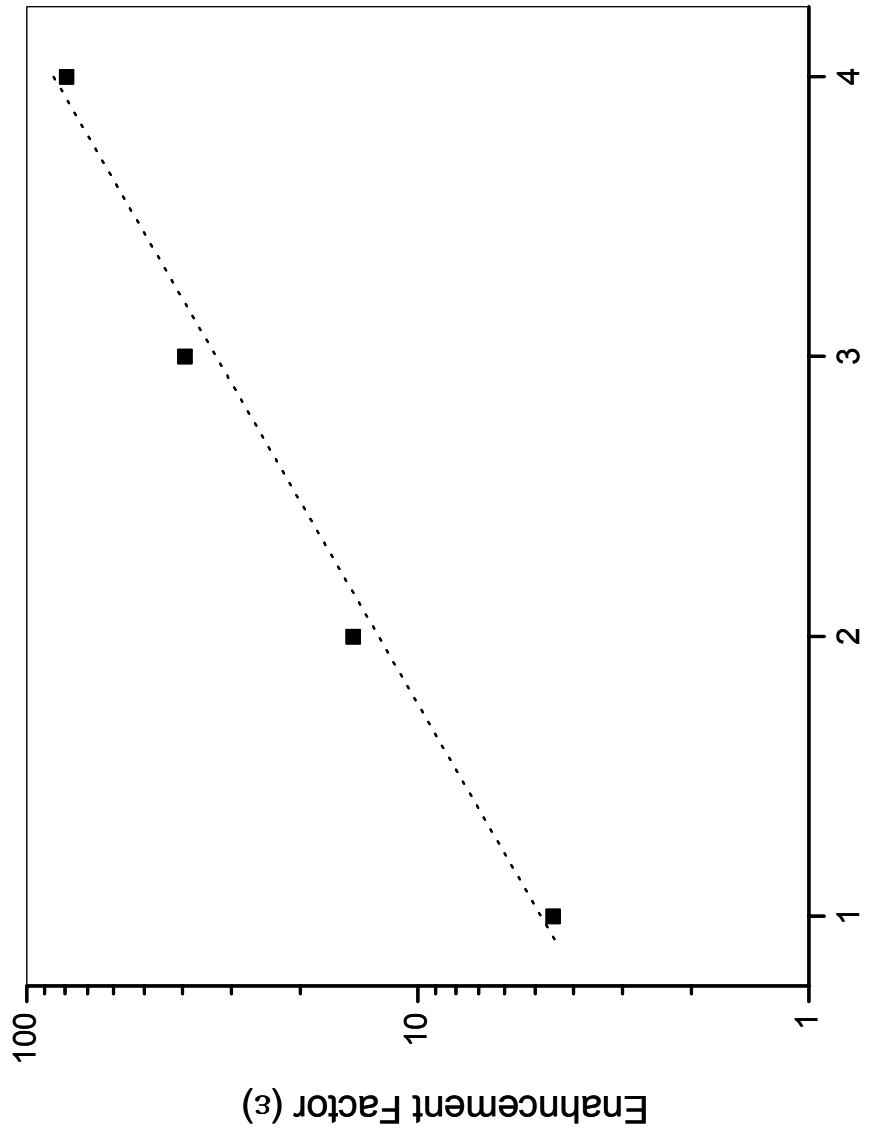


Fig. 5-7: Enhancement factor, for yields of Phe $[M-H]^-$ from bombardment by 15 keV/atom Au_3^+ and Au^+, as a function of the number of secondary ions detected per event. Line is a guide for the eye.

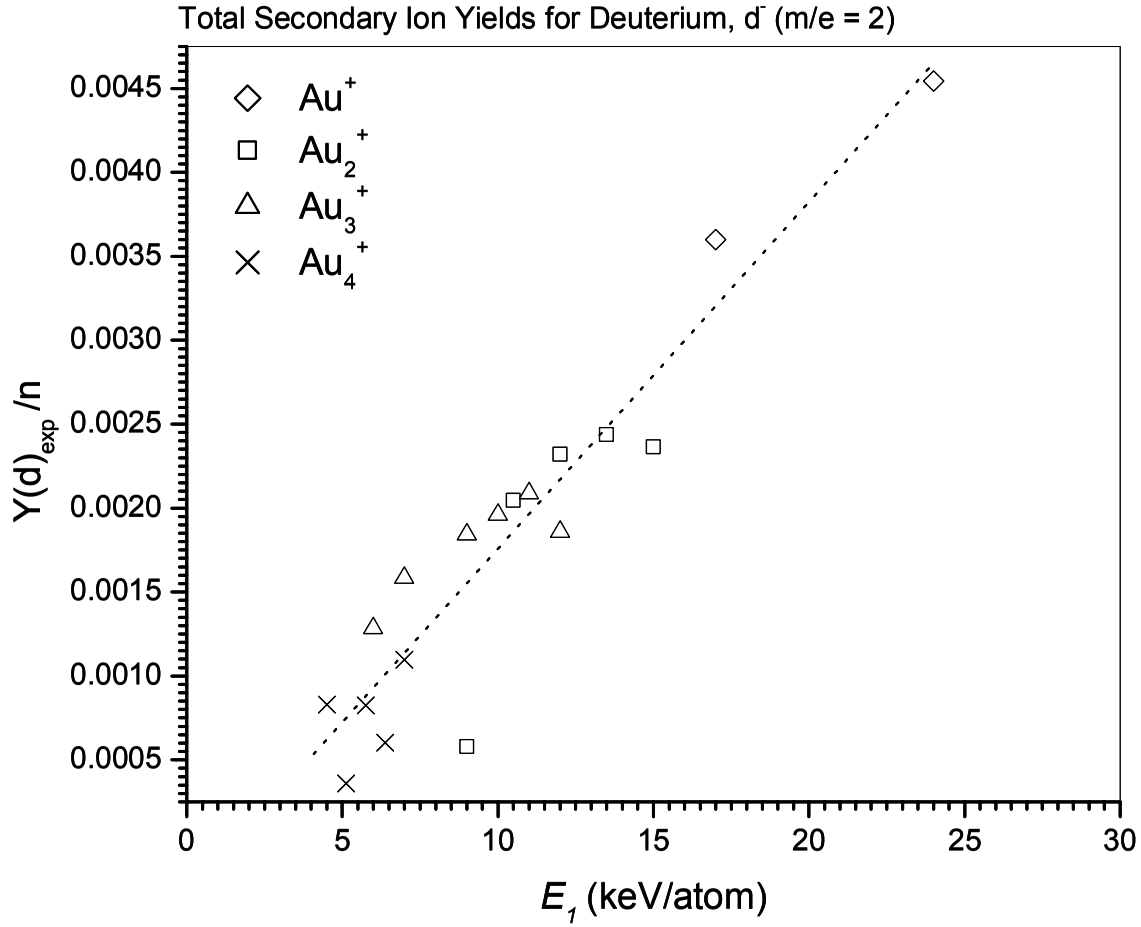


Fig. 5-8: Total secondary ion yields for the deuterium ion, D^- on a per atom basis as a function of the energy per atom of the Au_n^{m+} projectiles, n = 1 to 4, m=1,2. Line is a guide for the eye.

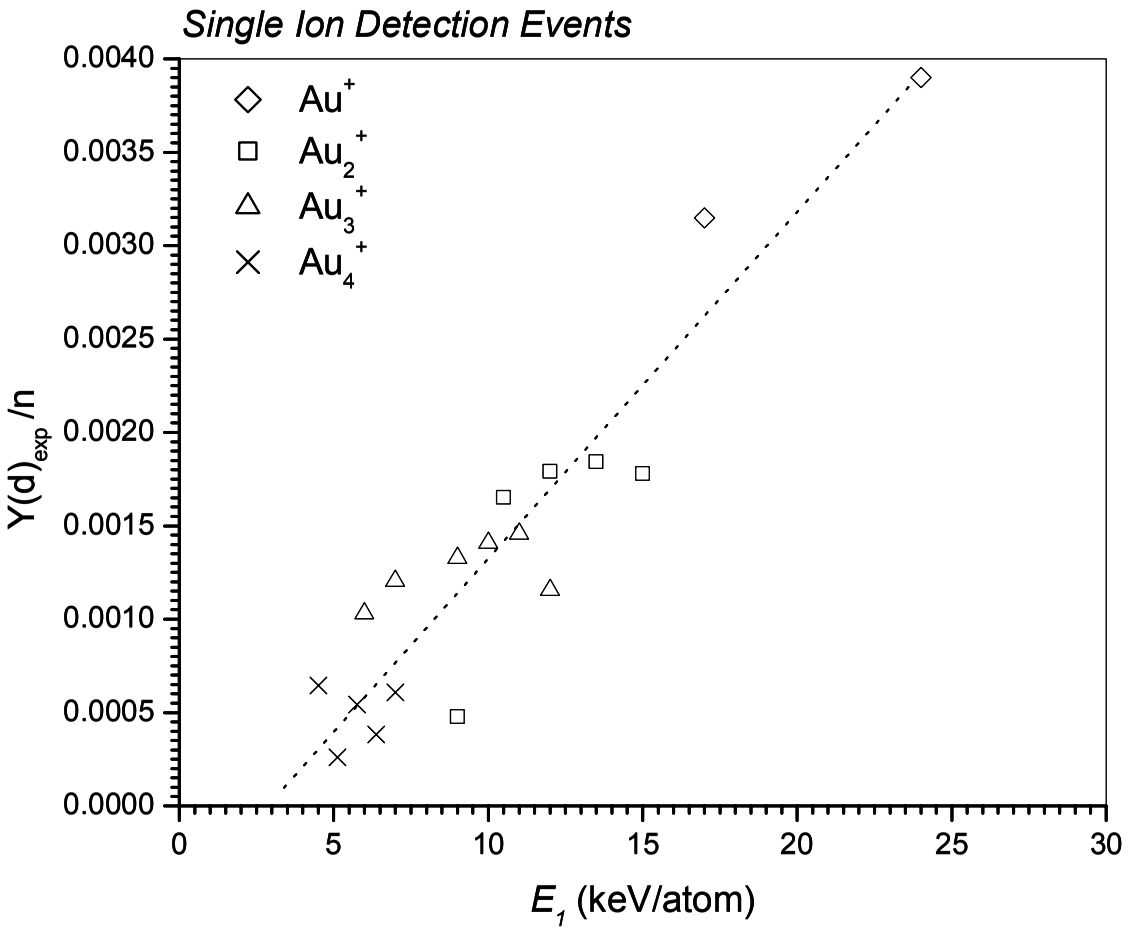


Fig. 5-9: Secondary ion yields for the deuterium ion, D^-, on a per atom basis as a function of the energy per atom of the Au_n^{m+} projectiles, n = 1 to 4, m=1,2 from one-ion detection events. Line is a guide for the eye.

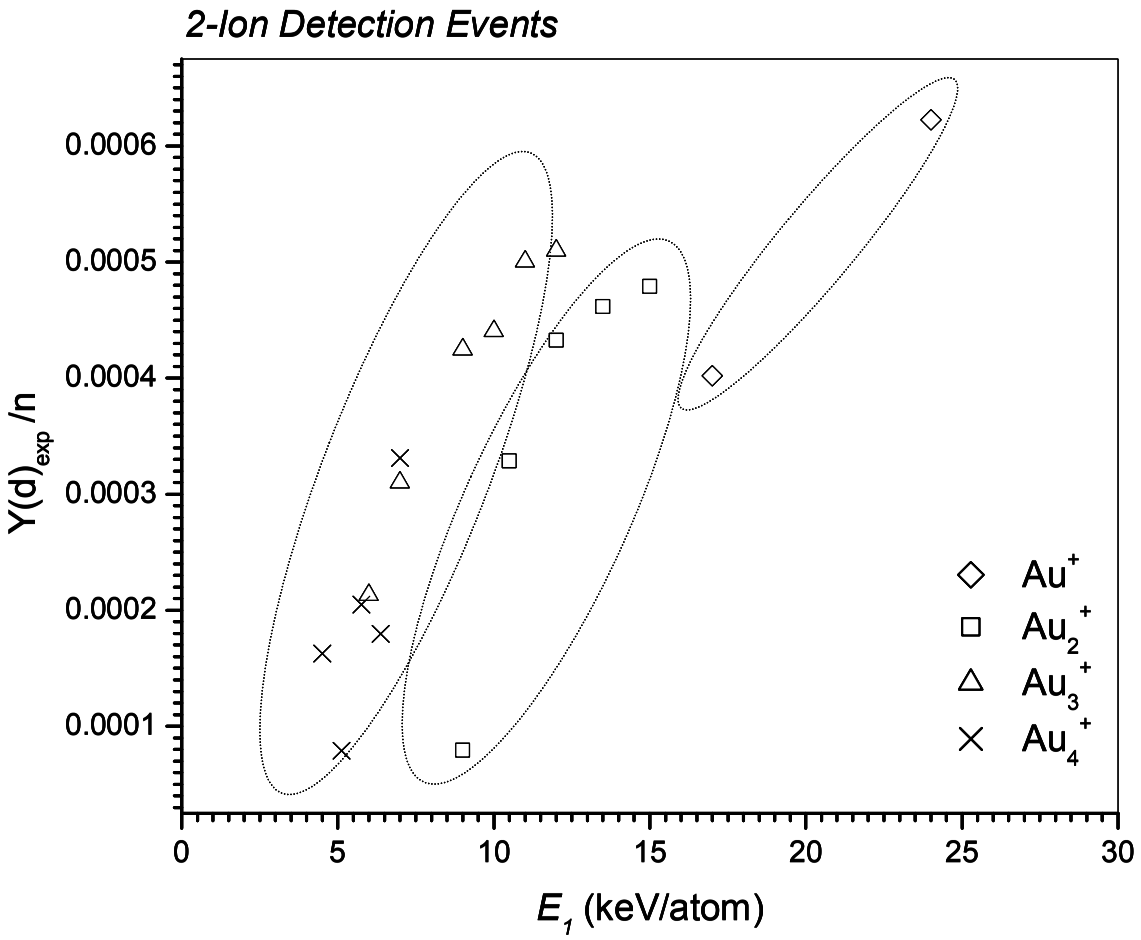


Fig. 5-10: Secondary ion yields for the deuterium ion, D^-, on a per atom basis as a function of the energy per atom of the Au_n^{m+} projectiles, n = 1 to 4, m=1,2 from two-ion detection events.

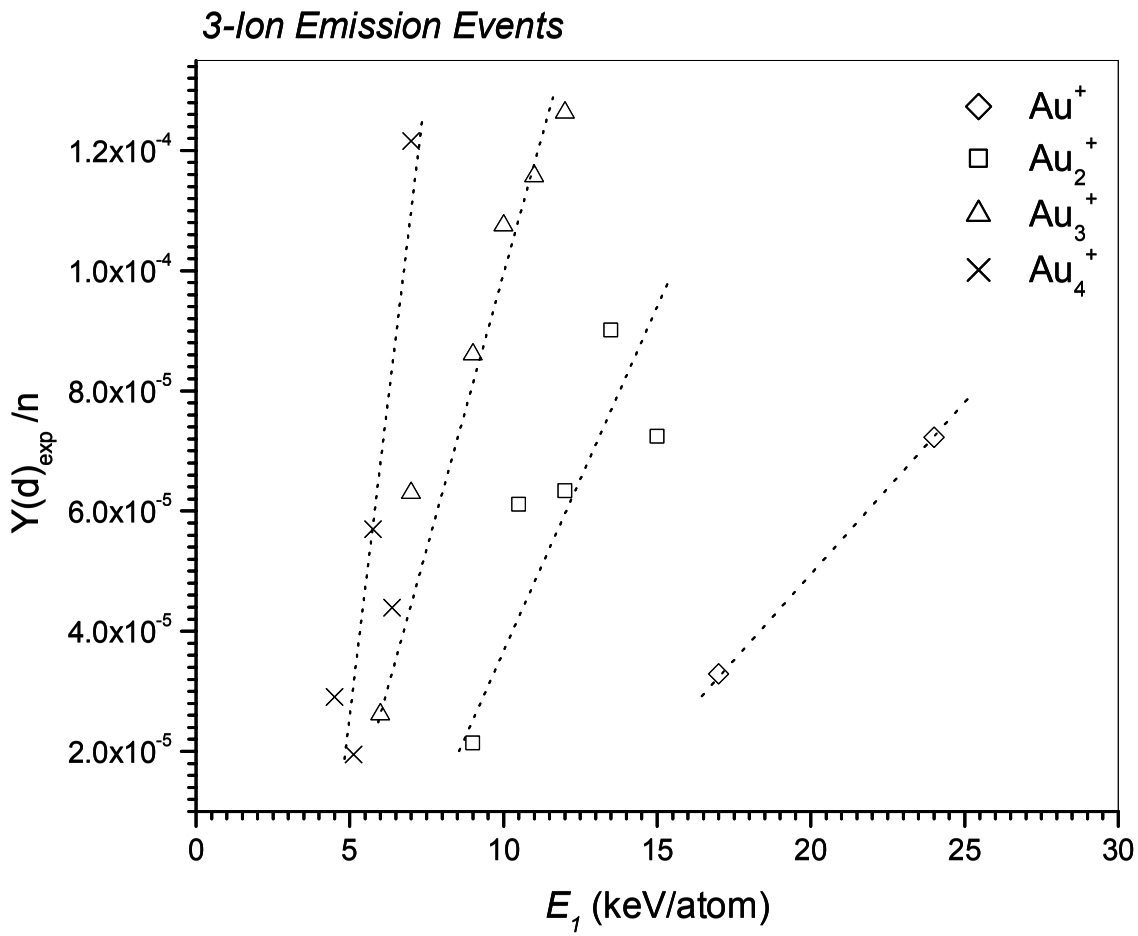


Fig. 5-11: Secondary ion yields for the deuterium, D^-, ion on a per atom basis as a function of the energy per atom of the Au_n^{m+} projectiles, n = 1 to 4, m=1,2 from three-ion detection events. Lines are a guide for the eye.

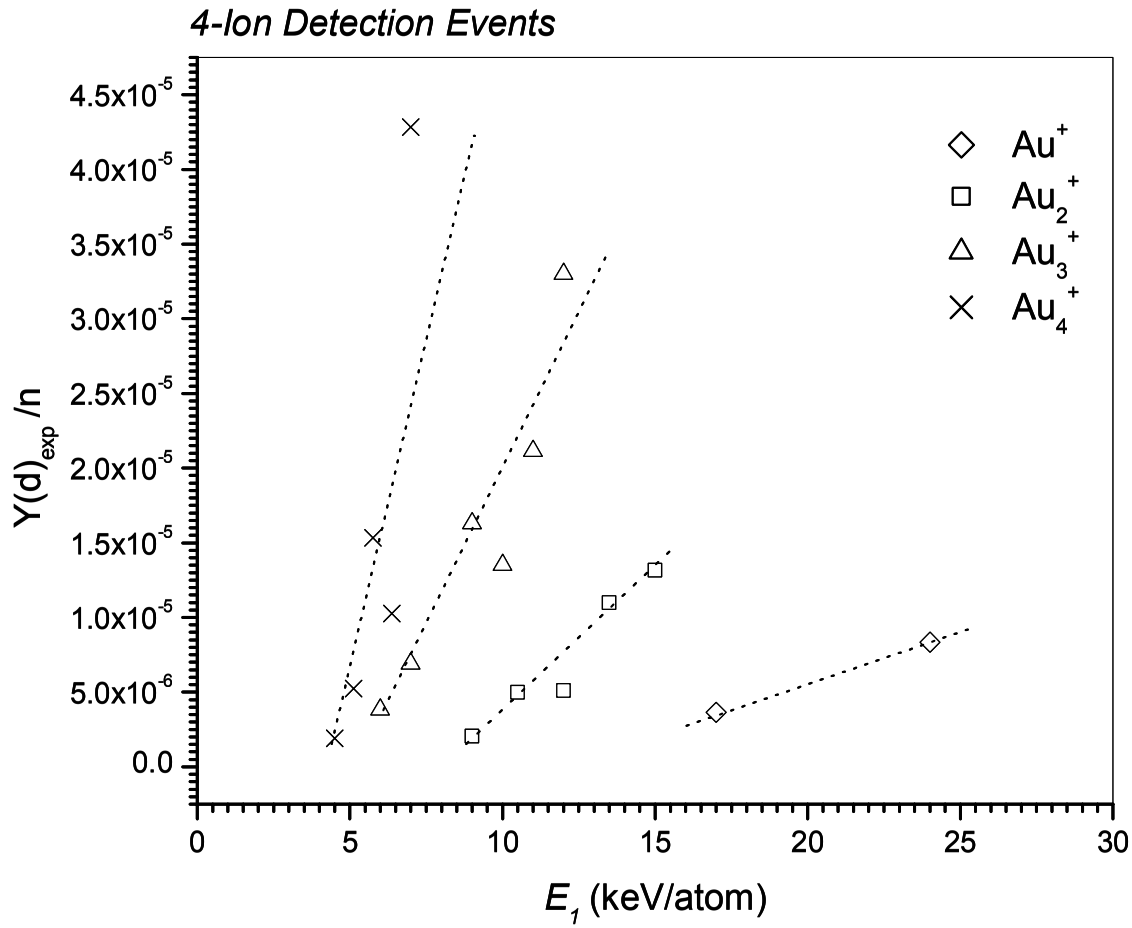


Fig. 5-12: Secondary ion yields for the deuterium ion, D^-, on a per atom basis as a function of the energy per atom of the Au_n^{m+} projectiles, n = 1 to 4, m=1,2 from four-ion detection events. Lines are a guide for the eye.

but has little influence in the overall yields for the atomic secondary ion deuterium, Fig. 5-8.

CHAPTER VI

CO-EMISSION OF MOLECULAR IONS FROM keV GOLD ATOMIC AND POLYATOMIC PROJECTILE IMPACTS*

Multiple emission events are of relevance for nano-analysis only if there is co-emission of analytically significant secondary ions. In Chapter V we examined multiple ion emission events as a function of projectile characteristics. We demonstrated that although most events were null with respect to secondary ion emission, there were cases where two or more secondary ions were detected. We also illustrated that the productivity of these events, measured by the enhancement factor, increases in a non-linear fashion. The occurrence of these "super-efficient" events raises two questions. First, what is their rate of production as a function of projectile characteristics? Further one must wonder if such co-emissions are due to different pathways in desorption/ionization with respect to single ion emission or simply due to random occurrences.

These questions were addressed by investigating the simultaneous ejection of two molecular ions as a function of keV projectile characteristics. We present here the

---

yields for the co-emission of two phenylalanine, Ph, molecular ions, and for comparison those pertaining to the emission of a single Ph molecular ion are also measured. Also included are data on the emission yield of charged dimers, Ph_2, (condensation product of two molecules).

In our pulse counting scheme it is not possible to detect ion multiplicity, i.e. the simultaneous arrival of two ions with the same mass. To recognize the ejection of two molecular ions we chose a target that consisted of an equimolar mixture of l-Ph (Aldrich P1, 700-8, M_w = 165.19) and deuterated l-Ph (Aldrich 49, 014-8, M_w = 173.26). This allows for the detection of two phenylalanine molecular ions, Ph_H $[M-H]^-$ (m/e = 164) and Ph_D $[M-H]^-$ (m/e = 172) similar in mass with the same physical characteristics. A conventional Time-of-Flight, ToF, mass spectrum of negative ions from 25 keV Au_3^+ bombardment of a Ph target is shown on Figure 6-1a. The yield of the Ph molecules emitted per one n-atom projectile impact, $Y_n(\mathrm{Ph})$, can be calculated from the observed yield of deuterated molecules, $Y_n(\mathrm{Ph_D})$, by:

$$Y_n(\mathrm{Ph}) = Y_n(\mathrm{Ph_D})/\rho = I_n\left(\mathrm{Ph_D}\right)/\alpha N \rho \tau \qquad \textbf{Eq. 6-1}$$

where $I_n(\mathrm{Ph_D})$ is the measured number of emitted molecular ions, α is the ionization probability, N is the number of bombardment events, ρ is the relative concentration of $\mathrm{Ph_D}$ ($\rho = 0.5$ for our experiment), and τ is the instrument transmission and detection efficiency, constant for all measurements. An analogous expression can be written using the measured number of $\mathrm{Ph_H}$ molecular ions.

In Figure 6-1a, is highlighted an inset of the molecular dimer region. Three peaks are observed, resulting from the three possible combinations Ph_H - Ph_H, Ph_H - Ph_D, and Ph_D - Ph_D. Their relative intensities of 1:2:1 reflect a binomial distribution. This is direct evidence that the Ph_H and Ph_D molecules were homogeneously mixed on the surface. Figure 6-1b is a "coincidence" ToF mass spectrum of all secondary ions co-emitted with the Ph_H molecular ion. The spectrum in Figure 6-1b is the sum of all individual events in which the Ph_H molecular ion was detected. The coincidentally recorded peak of the Ph_D molecular ion indicates that two negatively charged intact molecular ions can be emitted simultaneously, as a result of a single polyatomic projectile impact. The intensity of Ph_D peak represents the number of co-emitted Ph_H and Ph_D molecular ions.

Figure 6-2 shows the yield of Ph molecular ions emitted per projectile. These data confirm previous measurements in this energy range with these types of projectiles [30, 31]. Figure 6-3 is a similar plot but now for the yields of two Ph molecular ions (Ph_H and Ph_D) co-emitted from a single projectile impact. Figure 6-4 is a plot of the secondary ion yields for the Ph dimer.

The yields can be considered from two viewpoints, the effectiveness of the projectile, and the question of correlation in the coincidental emission. The effectiveness of an n-atom projectile over the atomic projectile can be expressed in the form of an enhancement factor ε (Eq. 2-1). As is shown on Figure 6-3 and 6-4, the yields Y_n depend on both the projectile energy per atom, $E_1 = E/n$ and the number of cluster constituents, n.

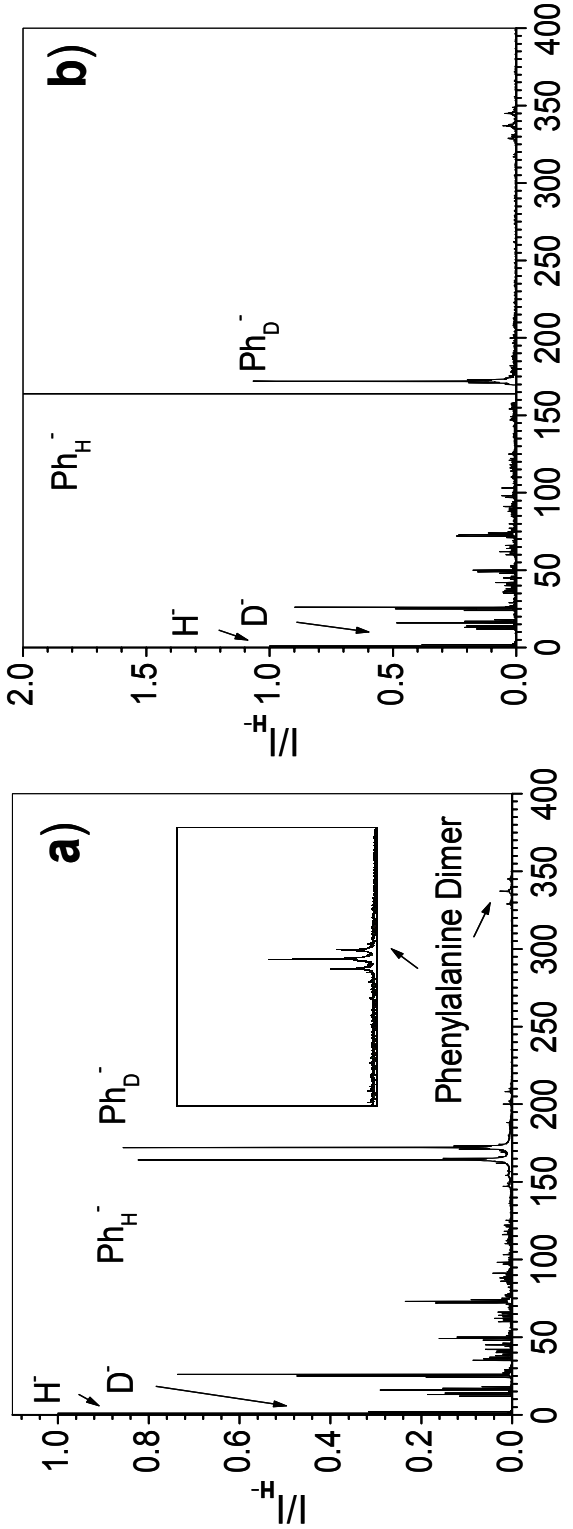


Figure 6-1: (a) Negative ion ToF mass spectrum of a Ph_H and Ph_D mixture from 25 keV Au_3^+ bombardment. Inset is from the molecular dimer region. (b) Negative ion coincidence ToF mass spectrum of all ions co-emitted with m/e 164.

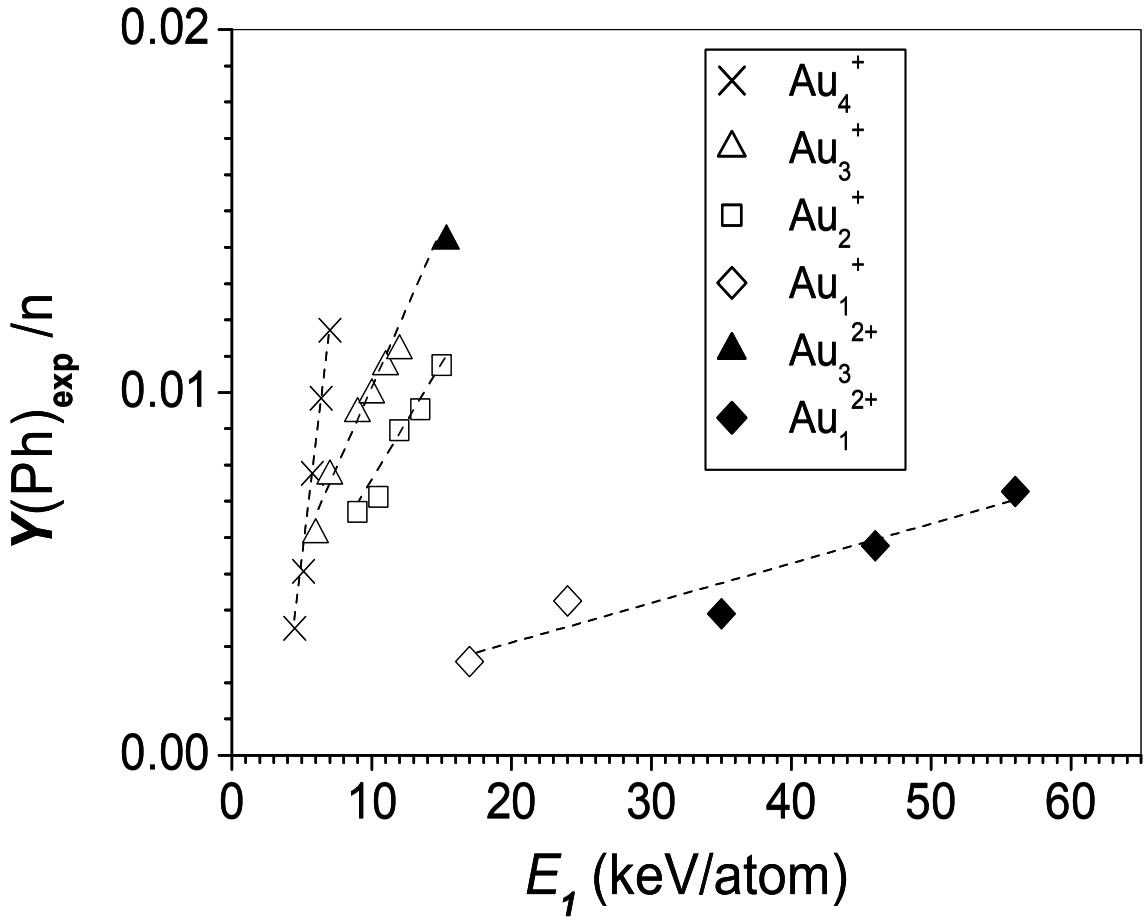


Figure 6-2: Secondary ion yields for the Ph molecular ion $(M-H)^-$ on a per atom basis as a function of the energy per atom of the Au_n^{m+} projectiles, n = 1 to 4, m=1,2. Accuracy in the experimental data is better than +/- 14% for the atomic projectiles and +/- 5% for the polyatomic projectiles.

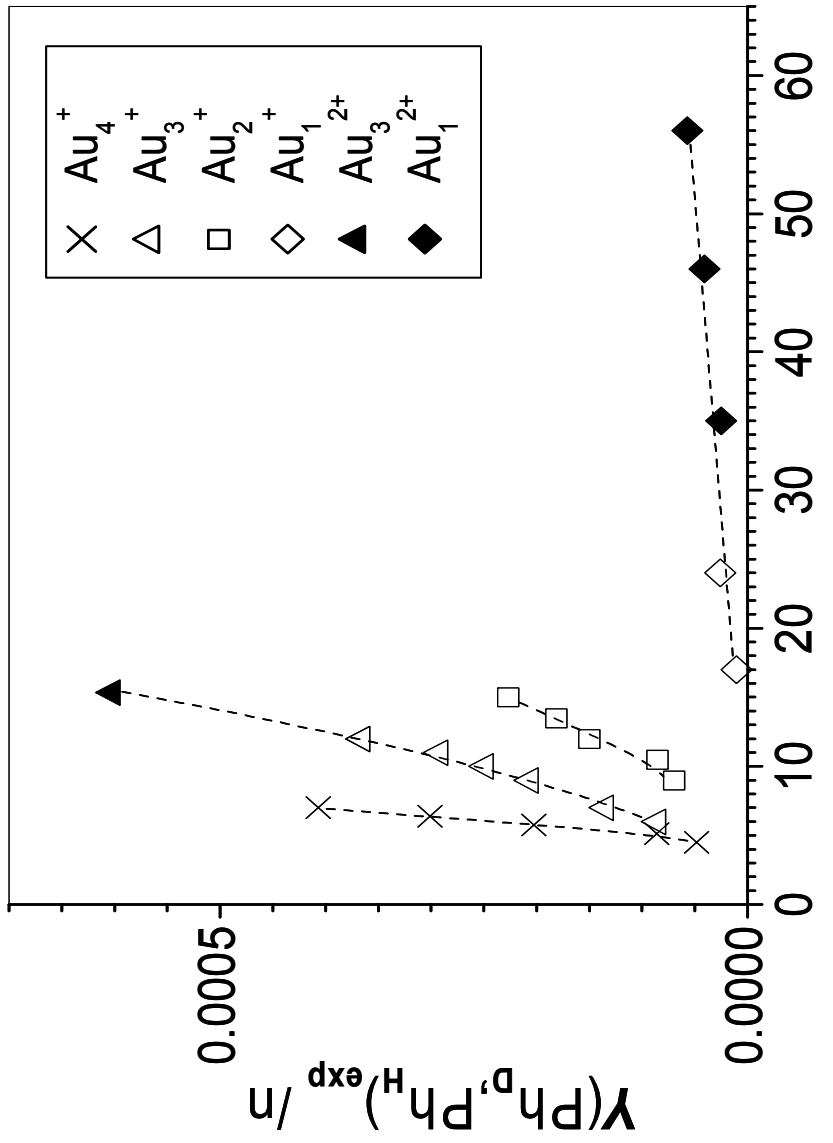


Figure 6-3: Secondary ion yields of co-emitted Ph_H and Ph_D molecular ions $(M\text{-}H)^-$ on a per atom basis as a function energy per atom of the Au_n^{m+} projectiles, n = 1 to 4, m=1,2. Accuracy in the experimental data is better than +/- 14% for the atomic projectiles and +/- 5% for the polyatomic projectiles.

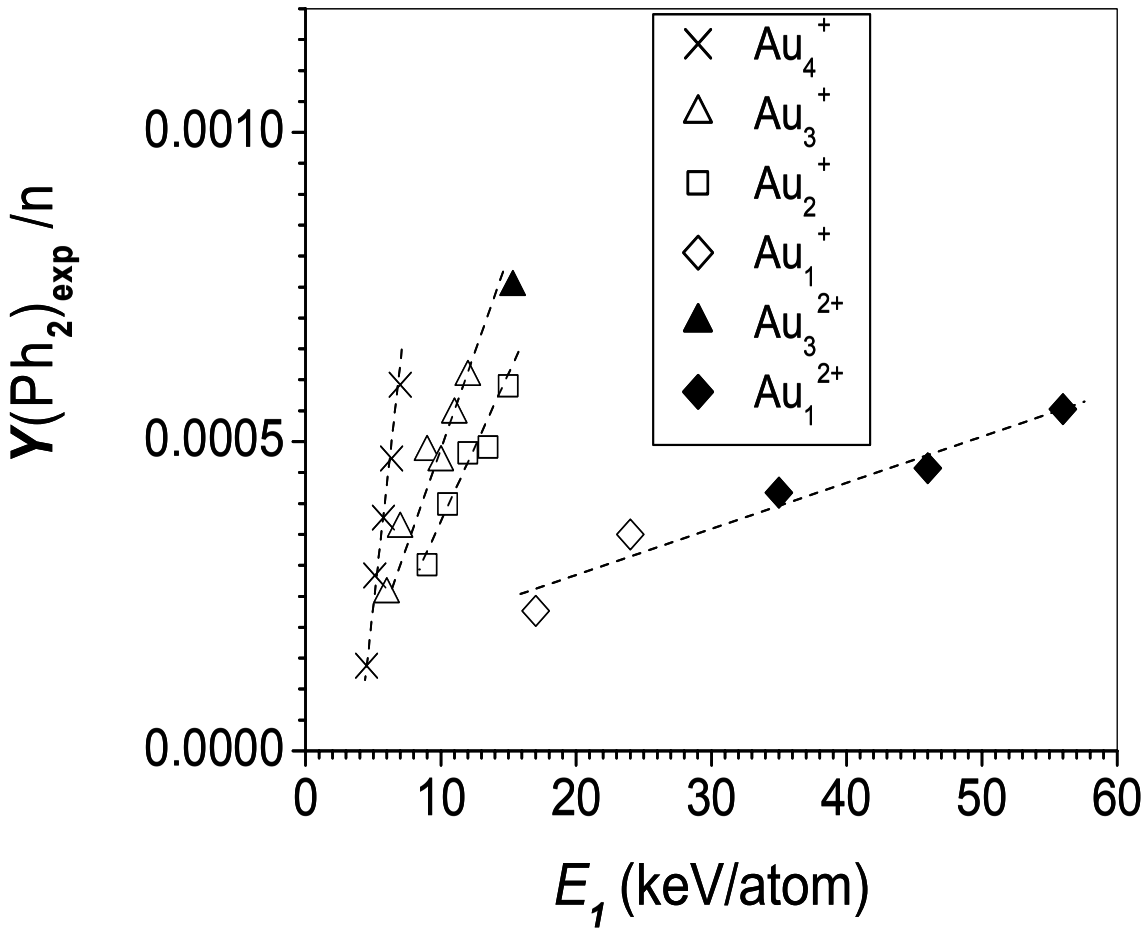


Figure 6-4: Secondary ion yields for the Ph dimer ion on a per atom basis as a function of the energy per atom of the Au_n^{m+} projectiles, n = 1 to 4, m=1,2. Accuracy in the experimental data is better than +/- 14% for the atomic projectiles and +/- 5% for the polyatomic projectiles.

The dependency of ε on these parameters can be expressed as follows: for the co-emission of two phenylalanine molecular ions $\varepsilon_{\mathrm{n}}\left(Ph_H, Ph_D\right) \propto \mathrm{n}^2 E_1^{2/3}$ and for the emission of the phenylalanine dimer $\varepsilon_{\mathrm{n}}\left(Ph_2\right) \propto \mathrm{n}^2 E_1^{4/3}$. The relatively strong dependence of $\varepsilon_{\mathrm{n}}\left(\mathrm{Ph}_2\right)$ on energy is caused by a different energy scaling of $Y_1\left(\mathrm{Ph}_2\right)$. This indicates that the pathway for emission of the dimer is different from that of molecular ion emission, involving perhaps the emission of more complex molecular assemblies that decay to the stable dimer configuration. For reference, the enhancement factor for the emission of single phenylalanine ions, $\varepsilon_{\mathrm{n}}\left(\mathrm{Ph}\right)$, shows an energy dependence of $E_1^{1/3}$ and, within the parameters explored here, is greater than unity, i.e. there is a "cluster effect". In the case for the emission of coincidental molecular ions and that of dimers, the respective values of ε_{n} are again greater than unity. More importantly, their respective scaling to E_1 shows a magnified "cluster effect". This means that those emissions are due to the overlapping of higher energy collision cascades and the creation of spikes which have been shown to be efficient for the emission of large molecules and their dimers [46, 47].

To determine if the molecular ion co-emission is correlated a correlation coefficient Q_{n} [48] can be calculated. First the relationship between the number of emitted molecular ions and the parameters describing the target composition must be established. The case presented here is for a two-component target and for event-by-event bombardment/detection. Each observation relates to the number of molecules emitted per event, *k*, and is small enough that the binomial distribution of Ph_H and Ph_D

molecules must be taken into account. In *k* emitted molecules there will be *i* molecules of Ph_D. The yield of deuterated molecular ions emitted is expressed as follows:

$$Y_n(Ph_D) = \sum_k \sum_{i=0}^{k} P_k i \Psi_i = (\sum_k k P_k)(\sum_{i=0}^{k} \frac{i}{k} \Psi_i) \qquad \textbf{Eq. 6-2}$$

In the above equation P_k is the probability that *k* molecules are emitted, and Ψ_i is the binomial distribution of Ph_D molecules within *k*. It may be recalled that Ψ_i is given by:

$\Psi_i = \frac{k!}{i!(k-i)!} \rho^i (1-\rho)^{k-i}$, with $\sum_{i=0}^{k} \frac{i}{k} \Psi_i = \rho$. The sum $\sum_k k P_k = Y_n(Ph)$ is the total sputtering yield of phenylalanine molecules. Thus, the yield of deuterated, molecular ions $Y_n(Ph_D) = Y_n(Ph_H) = \rho Y_n(Ph)$ regardless of the distributions of Ψ_i and P_k.

The situation is different when two phenylalanine molecules are co-emitted. In this case, for any given impact the relative abundances of Ph_H and Ph_D fluctuate within the random number of *k* emitted molecules. The yield of co-emitted Ph molecules $Y_n(Ph,Ph)$ can be calculated from the measured co-emission yields $Y_n(Ph_H,Ph_D)$ as follows:

$$Y_n(Ph,Ph) = Y_n(Ph_H,Ph_D)/\rho^2 = I_n(Ph_H,Ph_D)/\alpha^2 N \tau^2 \rho^2 = \sum_k \sum_{i=0}^{k} (k-1)(i - i^2/k) P_k \Psi_i / \rho^2 \qquad \textbf{Eq. 6.3}$$

where $I_n(Ph_H,Ph_D)$ is half the measured number of simultaneously emitted Ph_H and Ph_D molecular ions as a function of Ψ_i and P_k. One should keep in mind that the detection technique does not discriminate between the co-emission events (Ph_H,Ph_D)

and $(\mathrm{Ph_D},\mathrm{Ph_H})$, so $I_\mathrm{n}(\mathrm{Ph_H},\mathrm{Ph_D})$ represents half of the total number of the detected coincidence events$(\rho=0.5)$.

As can be seen from Figure 6-3, the energy of the projectile and the number of constituent atoms affect the yields. We can compare the yields of the single and co-emitted molecules using an approach from Ref. 48. The degree of correlation in the yields of co-emitted molecular ions can be described by the experimentally derived correlation coefficient $Q_\mathrm{n}(\mathrm{Ph_H},\mathrm{Ph_D})$ [6] defined as:

$$Q_\mathrm{n}(\mathrm{Ph},\mathrm{Ph}) = \frac{Y_\mathrm{n}(\mathrm{Ph},\mathrm{Ph})}{Y_\mathrm{n}(\mathrm{Ph})Y_\mathrm{n}(\mathrm{Ph})} = \frac{Y_\mathrm{n}(\mathrm{Ph_H},\mathrm{Ph_D})}{Y_\mathrm{n}(\mathrm{Ph_H})Y_\mathrm{n}(\mathrm{Ph_D})} \qquad \textbf{Eq. 6-4}$$

Substitutions from Eqs.1 and 3 give the expression of Eq.(6-4) in the form:

$$Q_\mathrm{n}(\mathrm{Ph},\mathrm{Ph}) = \frac{I_\mathrm{n}(\mathrm{Ph_H},\mathrm{Ph_D})}{I_\mathrm{n}(\mathrm{Ph_H})I_\mathrm{n}(\mathrm{Ph_D})}N \qquad \textbf{Eq. 6-5}$$

which shows that $Q_\mathrm{n}(\mathrm{Ph},\mathrm{Ph})$ does not depend on α,τ,ρ and can be calculated direct from the experimentally determined values $I_\mathrm{n}(\mathrm{Ph_H},\mathrm{Ph_D})$, $I_\mathrm{n}(\mathrm{Ph_H})$, $I_\mathrm{n}(\mathrm{Ph_D})$ and N.

A plot of experimental values of Q_n versus kinetic energy per projectile atom is shown in Figure 6-5. The experimental data can be compared with calculated ones on different yield distributions, P_k (Eqns. 6-2 – 6-4). The behavior of Q_n indicates that P_k is different for atomic and cluster projectiles. To our knowledge, information describing P_k for the sputtering of intact molecules does not exist in the literature. It may be mentioned that molecular dynamics simulations show qualitative differences in the sputtering yield distributions of silicon clusters by Au and Au_2 bombardment [49]. This

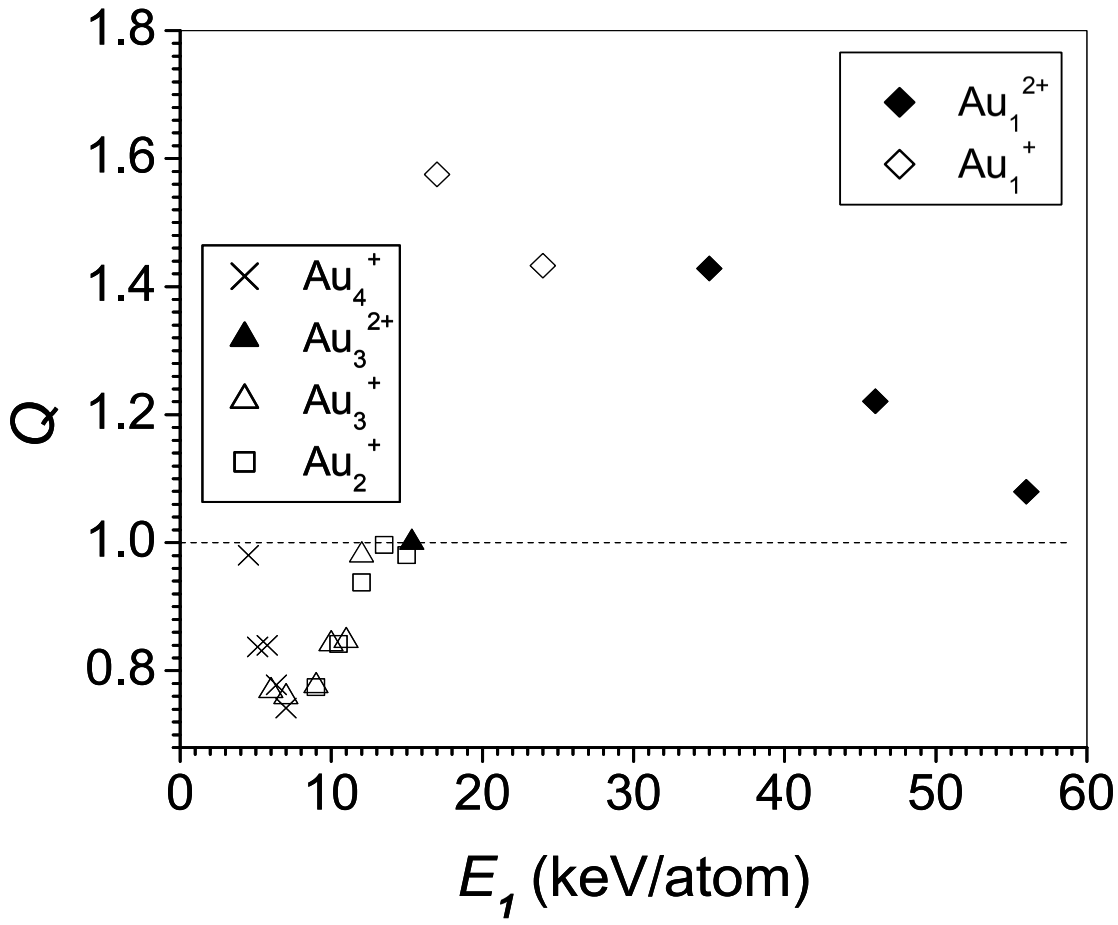


Figure 6-5: *Q* as a function of energy/atom for the selected projectiles for the case of the co-emission of two Ph molecular ions. Accuracy in the experimental data is better than +/- 14% for the atomic projectiles and +/- 5% for the polyatomic projectiles.

qualitative result indicates that P_k for polyatomic projectiles can be approximated by a symmetrical function, possibly a Poisson-like function. Indeed, the Poisson-like function used as P_k in Eq. (6-5) models a behavior of Q_{n} similar to our experimental curve.

For atomic projectiles a negative binomial distribution is an appropriate approximation for P_k. The behavior of the calculated Q_{n} is similar to the experimental function (Figure 6-5), when P_k is broad and asymmetric. Surprisingly, a similar distribution was obtained by computer simulation for the sputtering of Ni atoms by Xe^+ ion bombardment [50]. For atomic and polyatomic projectiles the behavior of P_k differs indicating different pathways of intact molecular ion emission.

The correlation coefficient Q_{n} (see Eqns. 6-4 & 6-5) allows for a detailed evaluation of the yields of co-emission of two molecular ions to those of single molecular ions by eliminating instrumental parameters such as transmission and detection efficiencies. At low bombardment energies (low coincidental ion yields), the different behavior of Q_{n} for atomic and polyatomic projectiles suggests that respective pathways for the emission of two molecular ions differ. For increasing energies of projectiles (increasing coincidental ion yields), Q_{n} tends to unity. The asymptotic approach to unity by Q_{n} for both cases (atomic and polyatomic projectiles) means that $Y_{\mathrm{n}}(\mathrm{Ph_H},\mathrm{Ph_D}) = Y_{\mathrm{n}}(\mathrm{Ph_D})Y_{\mathrm{n}}(\mathrm{Ph_H})$. The yield of simultaneously emitted molecules $Y_{\mathrm{n}}(\mathrm{Ph_H},\mathrm{Ph_D})$ is simply a product of the yields for emission of single intact molecules,

which means the absence of correlation. Indeed as the energy increases, yields increase and the influence of fluctuations in the relative densities of the Ph_H and Ph_D molecules on the value of $Y_n(Ph_H, Ph_D)$ decreases.

Our data show large enhancements in the yields of co-emitted molecular ions for more complex projectiles. For example, at ~ 15 keV/atom, the coincidental phenylalanine yields are 50 to 100 times larger for Au_2^+ and Au_3^+ respectively than for Au^+. The enhanced yields point to a concomitant increase in contribution from "super-efficient" collision cascades.

CHAPTER VII

CHARACTERIZATION OF SURFACE STRUCTURE BY CLUSTER COINCIDENTAL ION MASS SPECTROMETRY

Previous chapters have focused on the chemical nature i.e. atomic and molecular ion emission from, events in which single and multiple secondary ions are detected. One can ask if the secondary ion signal can also reveal differences in surface structure. This question is prompted by earlier PDMS experiments which have probed a compound successively in amorphous and crystalline states [15]. Significant differences in the mass spectra were observed albeit there was no change in the compound's stoichiometry. It was shown that negatively charged polyatomic secondary ions are ejected by the direct emission processes and are fragments representative of the surface structure [15]. This early work relied on Coincidence Counting Mass Spectrometry, CCMS [15]. In this methodology, one registers only secondary ions emitted in coincidence with a selected secondary ion. However recording secondary ions in this manner entails a measurement time dictated by the frequency of detecting the specific secondary ion. A more efficient approach is to record all events, i.e. all secondary ions detected from each impact. One can then in a subsequent offline data analysis seek coincidental ion emission.

In this study we apply the TME data acquisition scheme with cluster SIMS analysis of surface structures. These studies were carried out on specimens of α zirconium bis-(monohydrogen orthophosphate)monohydrate, α-ZrP, samples in the gel and crystalline states. α-ZrP, a clay-like material is used as an ion exchanger, consists of

Zr, P, O, and H atoms arranged in a structure that can range from an amorphous material to a well ordered crystalline material, making this compound well suited for this type of analysis. Targets were prepared by mixing 50mg of the ZrP into 1 mL ethanol and depositing 50 μL of the slurry onto a stainless steel target.

Figure 7-1 lists several fragment ions that can be traced back to the crystal structure of α-ZrP [15]. Negatively charged cluster secondary ions are the result of direct emission processes which suggests that they exist intact on the surface or result from the fragmentation of larger assemblies [15]. Thus, their intensities in the mass spectrum of the crystalline material should be higher than in the amorphous material.

Figure 7-2 is a mass spectrum of a ZrP gel sample with the high mass region expanded in the inset. The two predominant peaks resulting from the ZrP sample are 63 amu, (PO_2^-) and 79 amu, (PO_3^-). Most of the higher mass structure-specific fragments are in low abundance. This has been attributed to lack of long range ordering, on the scale of the emission area [51]. Figure 7-3 is a powder diffraction spectrum of the same sample. The peaks are broad suggesting some evidence of structure in the gel. This may explain why some structure-specific fragments are present in the mass spectrum of the gel sample, Fig. 7-2. Figure 7-4 is a mass spectrum of a crystalline ZrP sample. Again, the two predominant peaks are 63 amu and 79 amu. However, the structure-specific fragments are much more abundant (inset). The presence of these peaks can be attributed to the fact that the surface is ordered to a greater degree compared to the amorphous sample. A powder XRD spectrum, Fig. 7-5, further supports this observation. A question one can now ask is: Are these differences due to contributions

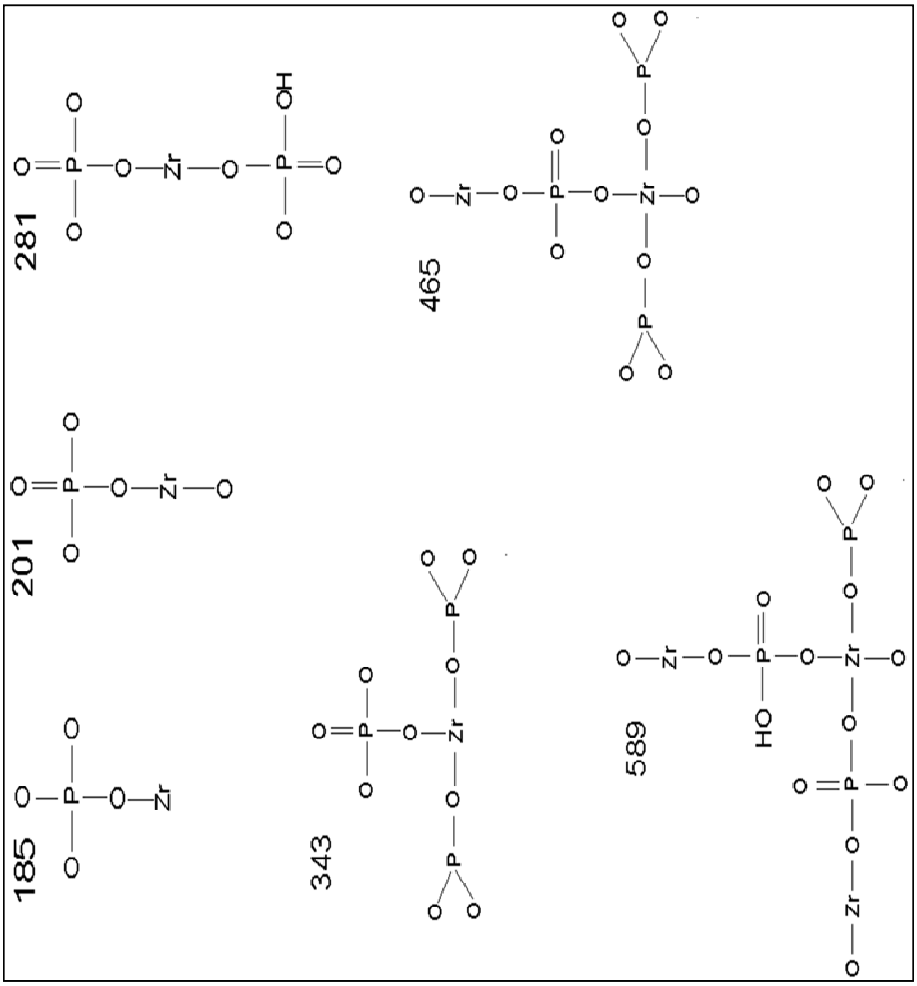


Fig. 7-1: Structure-specific fragments that can be traced back to the crystal structure of alpha-zirconium phosphate.

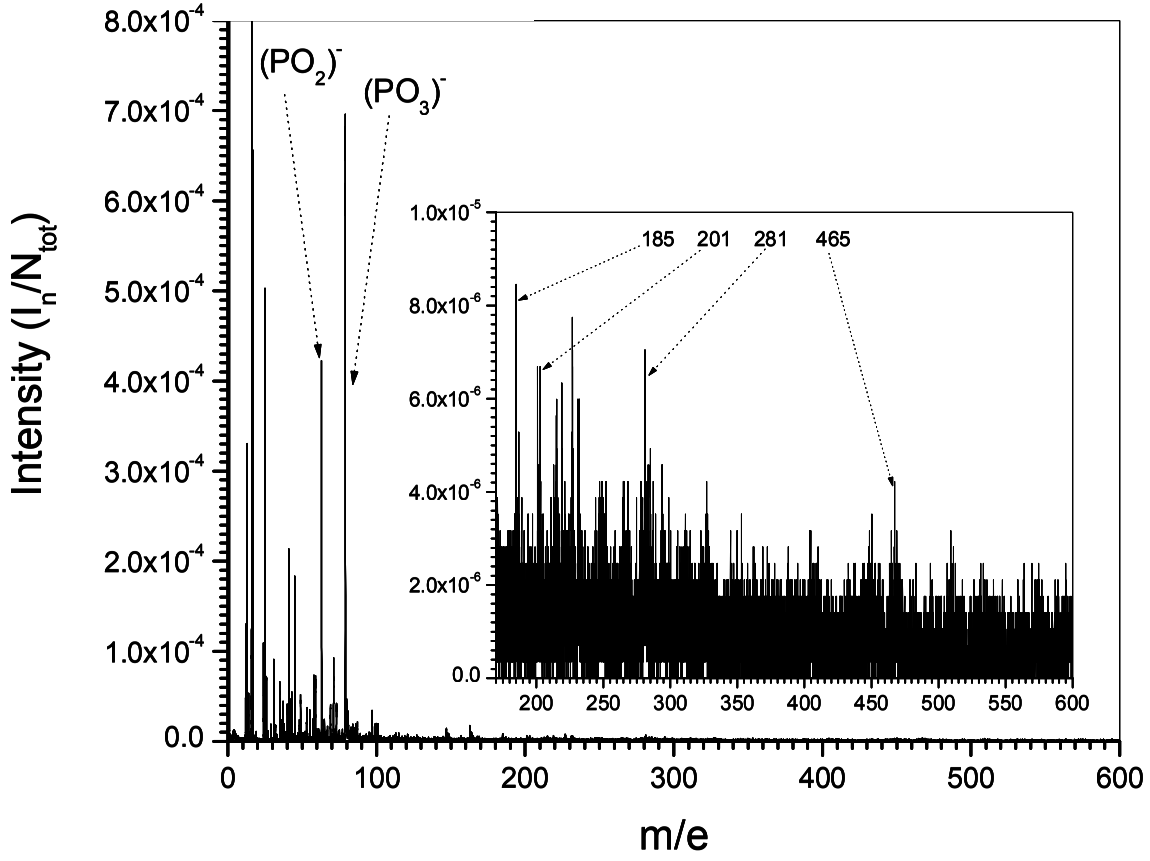


Fig. 7-2: Mass spectrum of a ZrP gel target from the impact of 22 keV Au_3^+ projectiles. Inset is an expansion of the 100 – 600 amu region.

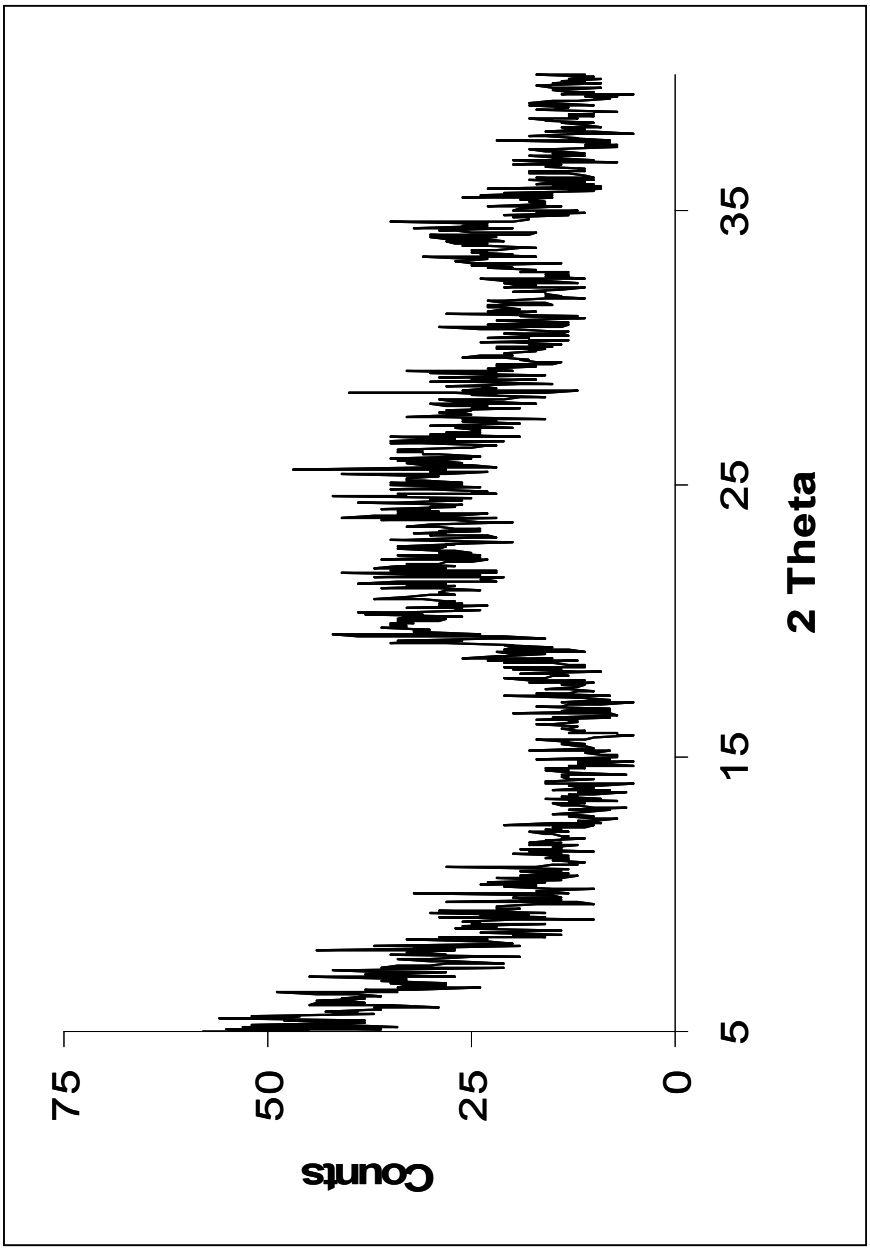


Fig. 7-3: Powder XRD spectrum of a ZrP gel material.

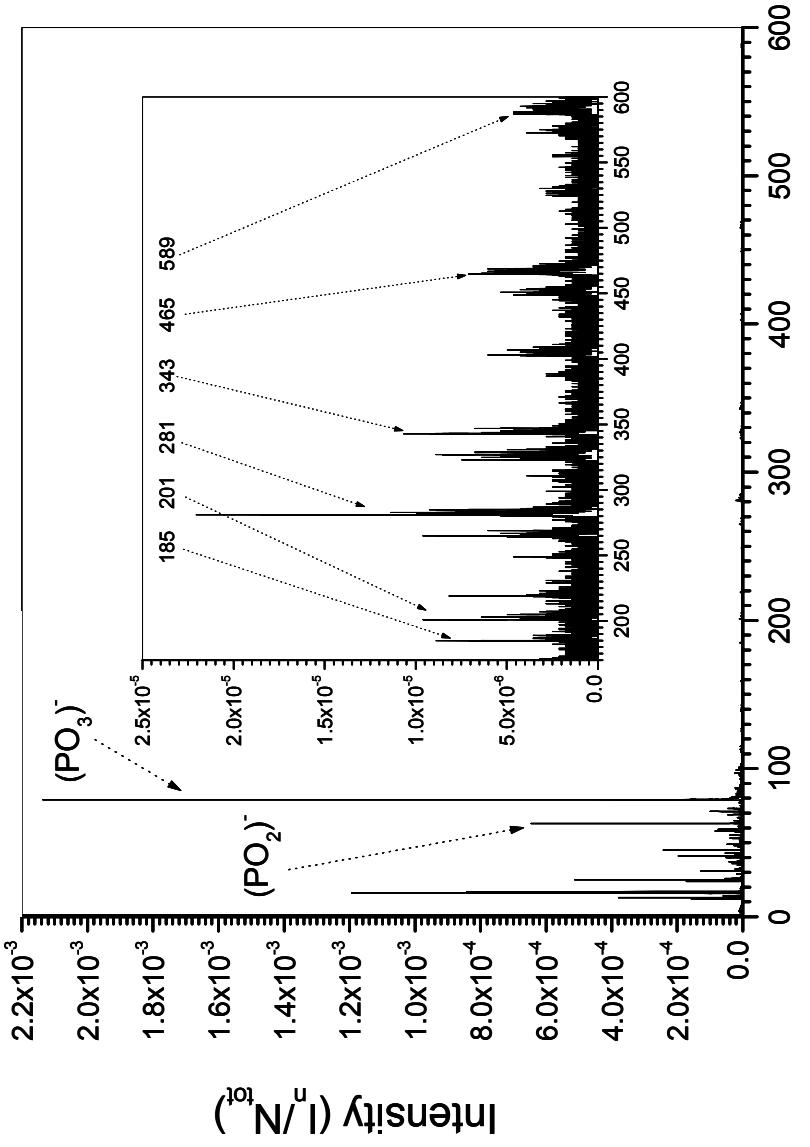


Fig. 7-4: Mass spectrum of a ZrP crystalline target from the impact of 22 keV Au_3^+ projectiles. Inset is an expansion of the 100 – 600 amu region.

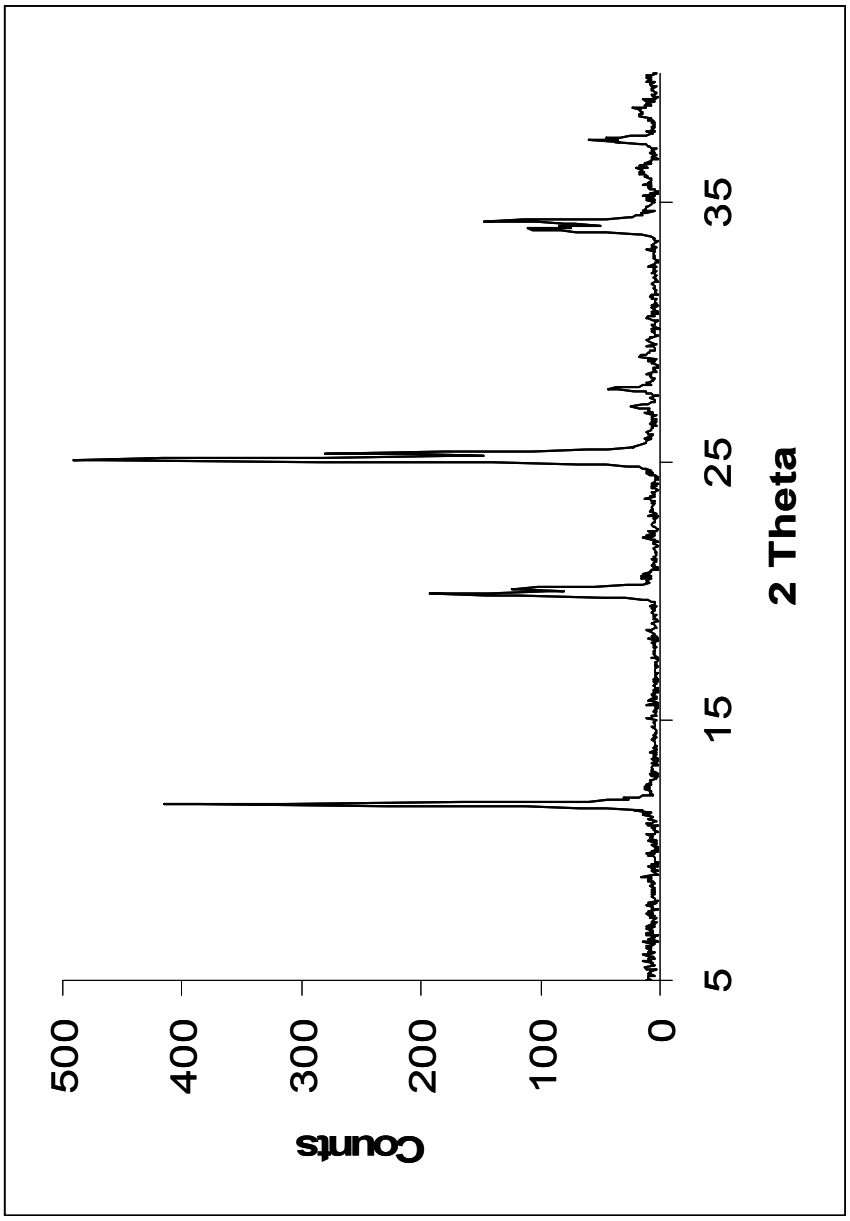


Fig. 7-5: Powder XRD spectrum of a ZrP crystalline material.

from events in which multiple secondary ions are ejected?

A methodology, similar to that employed to characterize multi-ion emission in Chapters V and VI, is used here. Figure 7-6 is a plot of the relative abundance of m/z=79 as a function of the total number of secondary ions ejected per single impact event. The relative abundance, R_a, is described here as:

$$R_a = \frac{I_{SI}^n}{I_{Total}^n} \qquad \textbf{Eq. 7-1}$$

where I_{SI}^n is the total number of a particular secondary ion, *SI*, from the sum of all *n*-ion events and I_{Total}^n are the total number of secondary ions from the sum of all *n*-ion events.

This ratio for PO_3^- in the gel sample is relatively constant. However, the ratio for PO_3^- decreases as the number of secondary ions per event increases in the crystalline sample. In the case of the crystalline sample, R_a for m/z=281 (Fig. 7-7) increases as the number of secondary ions ejected per event increases. This same trend is also present when looking at the fragment corresponding to m/z = 343, (Fig. 7-8). One explanation that can account for these trends is that secondary ions from multiple ion ejection events are less excited vibrationally. The vibrational energy deposited by the projectile may be more efficiently dissipated through a larger crystal lattice network resulting in a fragment ion that is more stable. Indeed as the number of secondary ions ejected per event increases, there is an increase in R_a for higher mass fragments, (Figs. 7-6 & 7-7), along with a concomitant decrease in R_a for PO_3^- (Fig. 7-5), for the crystalline material. In the amorphous sample the vibrational energy is imparted to a disordered system with less energy dissipation resulting in more fragmentation. This is apparent when

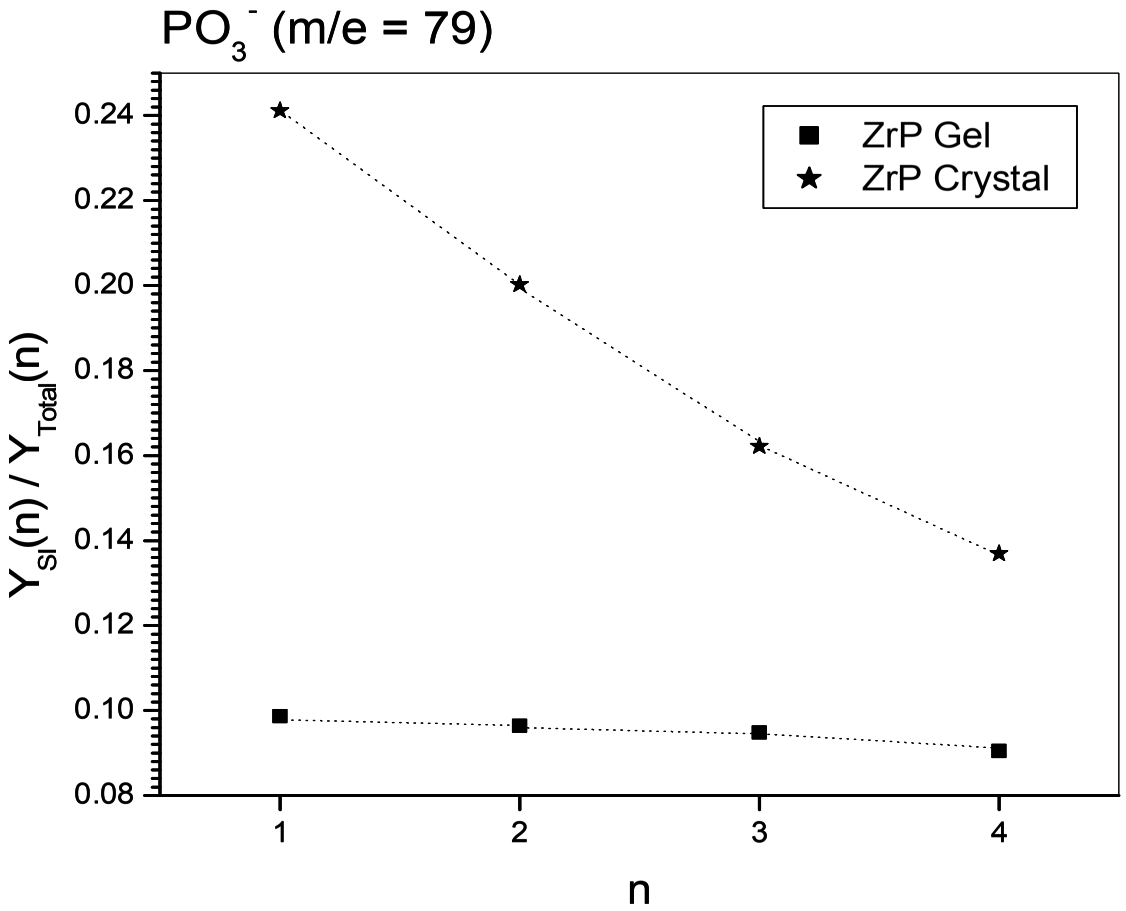


Fig. 7-6: The yield of m/z=79 (PO_3^-) as a function of the total number of secondary ions ejected per single impact event. Lines are a guide for the eye.

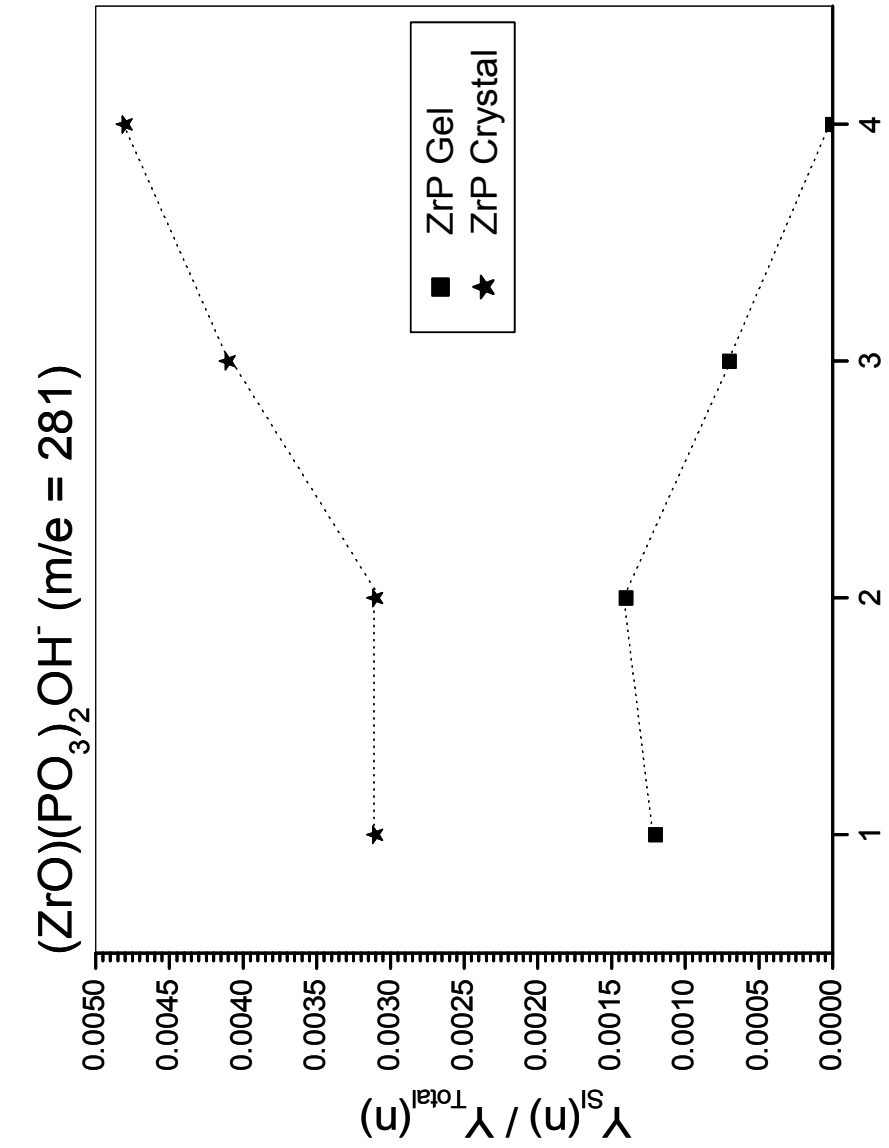


Fig. 7-7: The yield of m/*z*=281 as a function of the total number of secondary ions ejected per single impact event. Lines are guides for the eye.

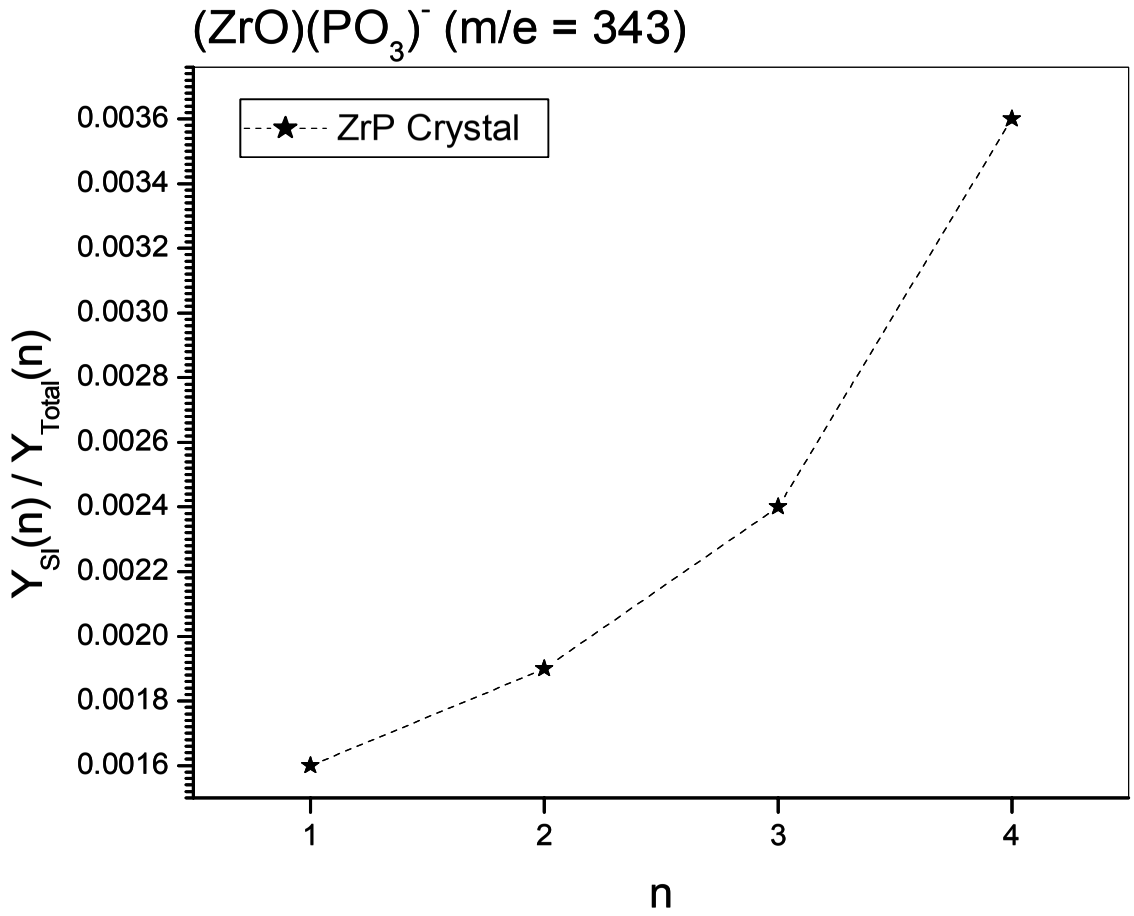


Fig. 7-8: The yield of m/z=343 as a function of the total number of secondary ions ejected per single impact event. Lines are guides for the eye.

considering the increase in R_a for m/z=281, Fig 7-6, and the relatively constant value of R_a for PO_3^-, Fig. 7-5 for the amorphous material.

We have qualitatively demonstrated that cluster SIMS can reveal changes in surface structure in the absence of any change in the analyte stoichiometry. Furthermore these results demonstrate that monitoring selected emission events, e.g. events when three or four secondary ions are ejected, can be very sensitive to changes in the surface structure.

CHAPTER VIII

CONCLUSIONS

The experiments run in the event-by-event bombardment/detection mode coupled with TME data archiving allowed us to examine cluster-solid interactions at the level of individual collision cascades. With this approach we could for the first time identify experimentally the "super-efficient" collision cascade. It had been postulated in simulations that these individual mega-events contribute disproportionately to the average sputter yield [52]. We have experimental confirmation that the "super-efficient" collision cascades, i.e. those responsible for the emission of multiple secondary ions, are responsible for the supra-linear enhancements of secondary ion yields in cluster SIMS.

We have investigated the occurrence of mega-events as a function of projectile characteristics *and* as a function of the number of secondary ions detected per event. It is readily apparent that some collision cascades are more efficient that others. Trends show an increasing degree of supra-linearity as the number of secondary ions detected per event increases. The enhancement factor for phenylalanine molecular ion emission from Au_4^+ bombardment in the single secondary ion detection events is ~ 5. The enhancement for the phenylalanine molecular ion in four secondary ion detection events is almost 70. Indeed one might term the four-ion event from the Au_4^+ projectile "super-efficient" for the molecular ion emission compared to that of the atomic projectile. Successful application of Coincidental Ion Mass Spectrometry, CIMS, rests with the ability of the primary ion to cause emission of multiple secondary ions from the same

event. Atomic projectiles offer similar yields to that of clusters. They are, however inefficient at inducing desorption of two or more secondary ions in a single event (Figs. 5-3 – 5-5). We also report for the first time to our knowledge the appearance of a "cluster-effect" for the emission of atomic secondary ions. Previous reported yields of atomic secondary ions as a function of projectile characteristics show the absence of this "cluster-effect". By monitoring multi-ion emission events we demonstrate that the previous trends are in fact a convolution of both linear and non-linear ion emission processes with the relative contribution of each to the observed yields dictated by the characteristics of the interrogating projectile. These results indicate that polyatomic projectiles are one requirement for the efficient application of CIMS.

A further topic addressed in this study is the effect of projectile characteristics on cases where two molecular ions are emitted from a single event. Our data show large enhancements in the yields of co-emitted molecular ions for more complex projectiles. For example, at ~ 15 keV/atom, the coincidental phenylalanine yields are 50 to 100 times larger for Au_2^+ and Au_3^+, respectively, than for Au^+. The enhanced yields point to a concomitant increase in contribution from "super-efficient" collision cascades. In addition a special case of molecular ion co-emission was observed, to wit the dimer which is the condensation product of two molecules. As in the case of molecular co-emission, the yields of the dimer as a function or projectile characteristics are enhanced when polyatomic projectiles are employed, however to a lesser degree, perhaps indicating that the pathways for emission of the dimer differ from those of the co-emission of two molecular ions.

The ZrP study provides an indication of the dependence of secondary ion emission on structural features. We have qualitatively demonstrated standpoint, that cluster CIMS can reveal changes in surface structure in the absence of any change in the analyte stoichiometry. Specifically, we show how the structure of the analyte can influence the appearance of the overall mass spectra. Further information is available when monitoring multi-ion emission events. We show how the relative yields of surface structure-specific fragments in these events depend on the surface ordering of the target.

Future studies in the area of cluster CIMS should focus on exploring if the trends for secondary ion yields as a function of projectile characteristics observed here exist for other analyte-specific secondary ions. The exploitation of multi-ion emission events has both practical and fundamental applications. For instance, in the case of structural characterization one could look for correlations between fragment secondary ions emitted within these types of events. This information may provide some insight into how the surface structure affects the emission and fragmentation of secondary ions. From a practical standpoint these types of events maximize the amount of chemical and physical information from nano-domains thus increasing the sensitivity over more conventional SIMS measurements. Ideally one wishes to increase the frequency of occurrence of multi-ion events. We have shown that polyatomic projectiles are more efficient in that regard compared to atomic projectiles. One wonders how even larger projectiles would perform.

Recent work with Au_{400}^{4+} suggests that "nano-particle SIMS" may be a powerful approach for efficient multi-ion emission. One might postulate that nano-particle SIMS

should be of interest for the characterization of nano-domains. This then prompts questions to be answered regarding the surface volume from which desorption occurs in the case of nano-particle bombardment.

A constant concern in SIMS is the analytical integrity of the signal. More specifically the relationship needs to be established between the nature, and especially the abundance, of secondary ions and the surface structure. A question yet to be addressed is the effect (if any) of the projectile characteristics on secondary ion emission for varying structural features.

Improved detection schemes must also be developed that can account for the detection of two secondary ions of the same mass if sensitivity is to be maximized. One possible solution is to use a multi-anode detector which can help eliminate this problem. There are however two drawbacks to using the multi-anode detector. The first problem is that a significant portion of the detection area (>20%) is lost due to the gaps that separate each individual anode. The second problem is that in order to detect two ions of the same mass, they must strike different anodes. If both strike the same anode then again they will be counted as a single secondary ion. A better solution would be to use a single anode with a method to record the pulse amplitude and correlate that to the number of ions that were responsible for it.

REFERENCES

[1] Evans Analytical Group, Sunnyvale, CA. URL: http://www.cea.com/literature/Bubble%20Chart.pdf, (Feb. 29, 2004).

[2] S. Chandra, D. R. Lorey, D. R. Smith, M. Miura, and G. H. Morrison, in *SIMS XII*, (Elsevier, Amsterdam 2000).

[3] C. Rollion-Bard, M. Chaussidon, C. France-Lanord, and E. Bard, in *SIMS XII*, (Elsevier, Amsterdam 2000).

[4] R. E. Peterson and B. J. Tyler, in *SIMS XII*, (Elsevier, Amsterdam 2000).

[5] A. P. Kovarsky, A. E. Nikolaev, and M. A. Jagovkina, in *SIMS XII*, (Elsevier, Amsterdam 2000).

[6] R. D. English, M. J. Van Stipdonk, and E. A. Schwekert, in *SIMS XII*, (Elsevier, Amsterdam 2000).

[7] D. Briggs and S. R. Bryan, in *SIMS XII*, (Elsevier, Amsterdam 2000).

[8] A. Benninghoven, F. G. Rudenauer, and H. W. Werner, in *Secondary Ion Mass Spectrometry – Basic Concepts, Instrumental Aspects, Applications and Trends,* (John Wiley and Sons, New York 1987).

[9] N. Winograd, Anal. Chem. **65,** 622A (1993).

[10] T. Kono, V. Vorsa, S. Sun, and N. Winograd, in *SIMS XII*, (Elsevier, Amsterdam 2000).

[11] W. Szymczak and K. Wittmack, Nucl. Instr. and Meth. in Phys. Res. **B88**, 149 (1994).

[12] S. V. Verkhoturov, E. A. Schweikert, and N. M. Rizkalla, Langmuir **18**, 8836, (2002).

[13] K. Wien, Lect. Notes Phys. **269,** 1 (1986).

[14] B. D. Cox, M. A. Park, R. G. Kaercher, and E. A. Schweikert, Anal. Chem. **64,** 843 (1992).

[15] M. VanStipdonk, M. A. Park, E. A. Schweikert, P. Sylvester, and A. Clearfield, Int. J. Mass Spectrom. Ion Processes **128,** 133 (1993).

[16] A. Benninghoven, B. Hagenhoff, and E. Niehuis, Anal. Chem. **65**, 630A (1993).

[17] R. Castaing and G. Slodzian, J. Micros. **1**, 395 (1962).

[18] R. F. K. Herzog and F. Viehboeck, Phys. Rev. **76**, 855 (1949).

[19] P. Sigmund, Phys. Rev. **184**, 383 (1969).

[20] P. Sigmund, Appl. Phys. Lett. **25**, 171 (1974).

[21] H. H. Anderson and H. L. Bay, J. Appl. Phys. **46**, 2416 (1975).

[22] I. S. Bitensky and E. S. Parilis, Nucl. Instrum. Methods Phys. Res. B **21**, 26 (1987).

[23] D. A. Thompson and S. S. Johar, Appl. Phys. Lett. **34**, 342 (1979).

[24] H. H. Anderson, Mat. Fys. Medd. Dan. Vidensk **43**, 127 (1993).

[25] Y. Le Beyec, Int. J. Mass Spect. and Ion Procc. **174**, 101 (1998).

[26] P. K. Rol, J. M. Fruit, and J. Kistemaker, Physica **26**, 1000 (1960).

[27] F. Grønlund and W. J. Moore, J. Chem. Phys. **32**, 1540 (1960).

[28] A. D. Appelhans, J. E. Delmore, and D. A. Dahl, Anal. Chem. **59**, 1685 (1987).

[29] A. D. Appelhans and J. E. Delmore, Anal. Chem. **61**, 1087 (1989).

[30] M. G. Blain, S. Della-Negra, H. Joret, Y. Le Beyec, and E. A. Schweikert, Phys. Rev. Lett. **63**, 1625 (1989).

[31] M. Benguerba, A. Brunelle, S. Delle-Negra, J. Depauw, H. Joret, Y. Le Beyec, M. G. Blain, E. A. Schweikert, G. Ben Assayag, and P. Sudraud, Nucl. Instr. and Method in Phys. Res. **B62**, 8 (1991).

[32] R. A. Zubarev, I. S. Bitensky, P. A. Demirev, and B. U. R. Sundqvist, Nucl. Instr. and Method in Phys. Res. **B88**, 143 (1994).

[33] J. H. Moore, C. C. Davis, and M. A. Coplan, in *Building Scientific Apparatus* (Perseus Books Pub., Reading, MA 1991).

[34] C. Dedman, E. Roberts, S. Gibson, and B. Lewis, Rev. Sci. Instr. **72**, 2915 (2001).

[35] J. A. Bennett, E. A. Schweikert, D. Poisson, and C. Joliceur, Surf. Int. Anal. **15**, 651 (1990).

[36] M. A. Park, K. A. Gibson, L. Quinones, and E. A. Schweikert, Science **248**, 988 (1990).

[37] I. S. Gilmore and M. P. Seah, Int. J. of Mass Spec. **202**, 217 (2000).

[38] J. L. Wiza, Nucl. Instr. and Meth. **162**, 587 (1979).

[39] D. A. Gedcke and W. J. McDonald, Nucl. Inst. and Meth. **58**, 253 (1968).

[40] M. A. Park, B. D. Cox, and E. A. Schweikert, J. Chem. Phys. **96**, 8171 (1992).

[41] S. Della-Negra. D. Jacquet, I. Lorthiois, and Y. Le Beyec, Int. J. Mass Spec. and Ion Phys. **53**, 215 (1983).

[42] W. Ens, K. G. Standing, and A. Verentchikov, in *Proc. Int. Conf. on Instr. for Time-of-Flight Mass Spec.* (LeCroy, Chestnut Ridge, NY 1992).

[43] R. D. Rickman, S. V. Verkhoturov, E. S. Parilis, and E. A. Schweikert, Phys. Rev. Lett. **92**, 47601 (2004).

[44] M. Benguerba, A. Brunelle, S. Della-Negra, J. Depauw, H. Joret, Y. Le Beyec, M. G. Blain, E. A. Schweikert, G. Ben Assayag, and P. Sudraud, IPNO-DRE-**91-16**, 1 (1991).

[45] R. Zaric, B. Pearson, K. D. Krantzman, and B. J. Garrison, Int. J. of Mass Spectrom. and Ion Processes **174**, 155 (1998).

[46] G. Gillen, and A. Fahey, Appl. Surf. Sci. **203-204**, 209 (2003).

[47] S. F. Belykh, I. S. Bitensky, D. Mullajanov, and U. Kh. Rasulev, Nucl. Instrum. Methods Phys. Res. B **129**, 451 (1997).

[48] K. B. Ray, M. A. Park, and E. A. Schweikert, Nucl. Instrum. Methods Phys. Res. B **82**, 317 (1993).

[49] M. Medvedeva, I. Wojciechowski, and B. J. Garrison, Appl. Surf. Sci. **203-204**, 148 (2003).

[50] W. Eckstein, Nucl. Instrum. Methods Phys. Res. B **33**, 489 (1988).

[51] M. J. VanStipdonk, J. B. Shapiro, and E. A. Schweikert, Vacuum **46** (8-10), 1227 (1995).

[52] M. H. Shaprio and T. A. Tombrello Surf. Sci. **453**, 143 (2000).

APPENDIX A

LIQUID METAL ION SOURCE FABRICATION

Read through this entire procedure before attempting to fabricate this source.

1.) To make spring reservoir:

   a. Cut a ~20 cm piece of tungsten wire (0.200 mm dia.). Clean with automotive grade sandpaper.

   b. Place center of wire on form (Fig A-1) and begin to wrap around the form with jerky motions toward previous turn. This breaks the memory of in the wire.

   c. Make 9 or 10 tight turns to form the spring reservoir (no less than 9). There should be no space between the individual turns,

2.) To make needle:

   a. Cut a section of tungsten wire, ~ 5 cm long, and clean with automotive grade sandpaper.

   b. Place the wire in the pin vise (Fig. A-2a). To break the memory in the wire, pull your fingers along the wire, from the vise to the tip of the wire, Fig. A-2b. Repeat this until the wire is straight.

   c. Cut the wire to a length of 3 cm.

3.) To make etching solution add:

   a. 50 ml glycerol

   b. 50 ml distilled water

   c. 10 ml 35% NaOH solution.

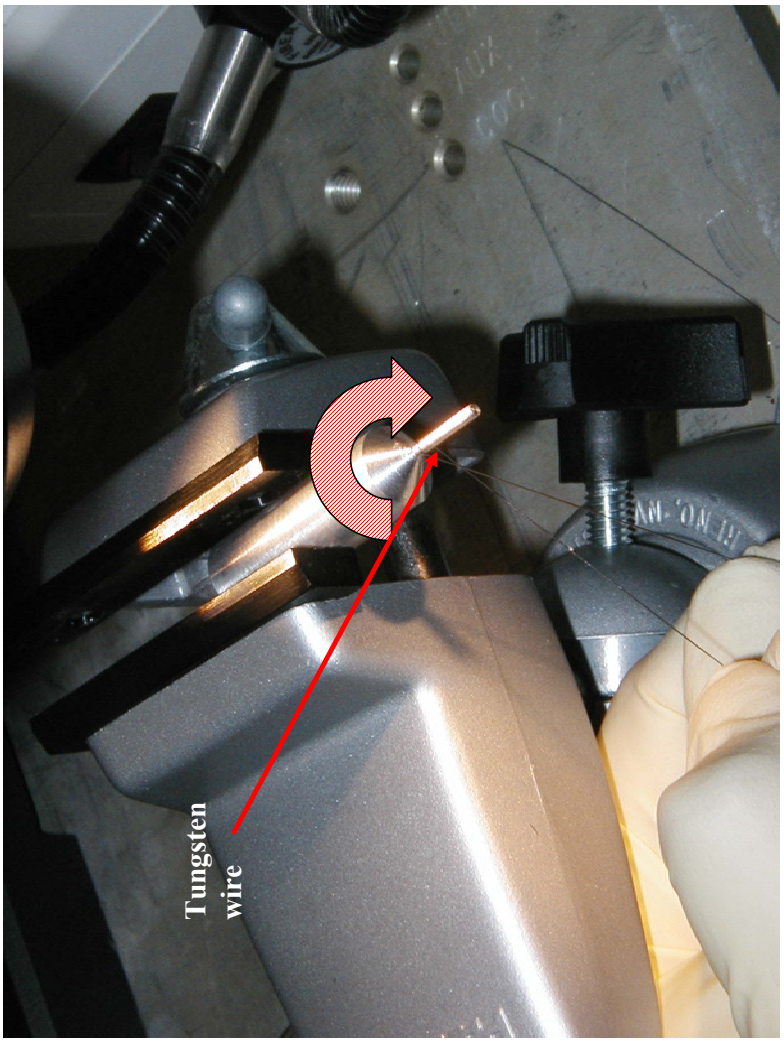
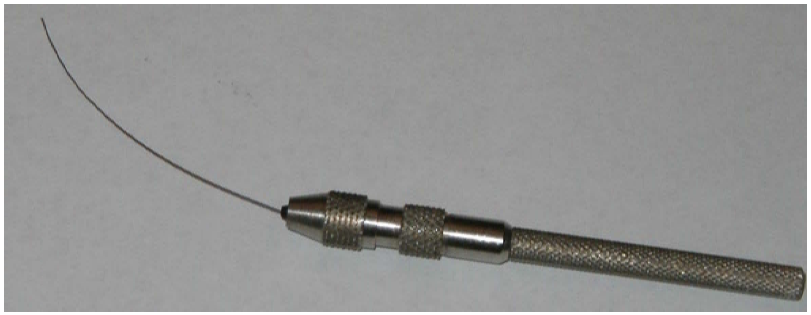


Fig. A-1: Making the reservoir.

a.)



b.)

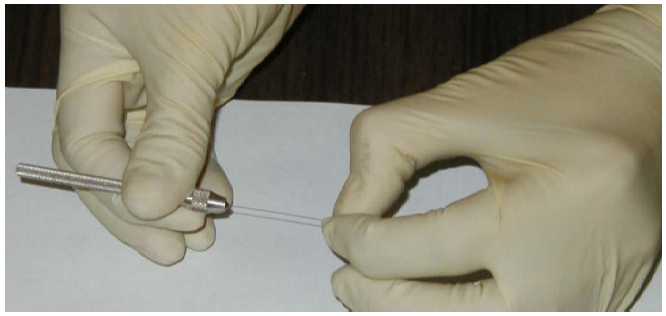


Fig. A-2: Needle fabrication.

d. and mix thoroughly.

4.) Etch the needle according to the following procedures.

a. The tip of the wire can be split as a result of cutting, Fig. A-3a. This must be removed before etching the needle. Attach electrical connections as shown in Fig. A-3b. Insert about 1mm of needle in the etching solution and turn the AC voltage to a high setting (~30 V). Remove the needle from the solution at 2 minute intervals and visually inspect, with the microscope. Repeat this procedure until the needle has a flush tip, Fig. A-3b.

b. While looking through the microscope, immerse the tip in the etching solution then pull it up just enough to maintain meniscus, Fig. A-4a. Apply a low AC voltage (~5-10 V) to form the cone at the tip of the needle. Raise the needle, at 5 minute intervals, to check cone formation. Repeat this procedure until a cone with a half angle of ~ 49.5° is formed, Fig. A-4b.

c. After the cone is formed reduce the AC voltage by half and immerse the needle 15 mm in the solution for ~ 10 minutes and remove from the solution

d. Rinse the needle with distilled water to remove remaining etching solution.

5.) Cut spring ends to proper length using the spring jig as a measure, Fig. A-5a. Insert the spring into the source assembly and tighten screws (use tweezers to hold spring leg to keep from twisting), Fig. A-5b. If spring is not horizontal use spring form to bring spring into a horizontal position, Fig. A-5c.

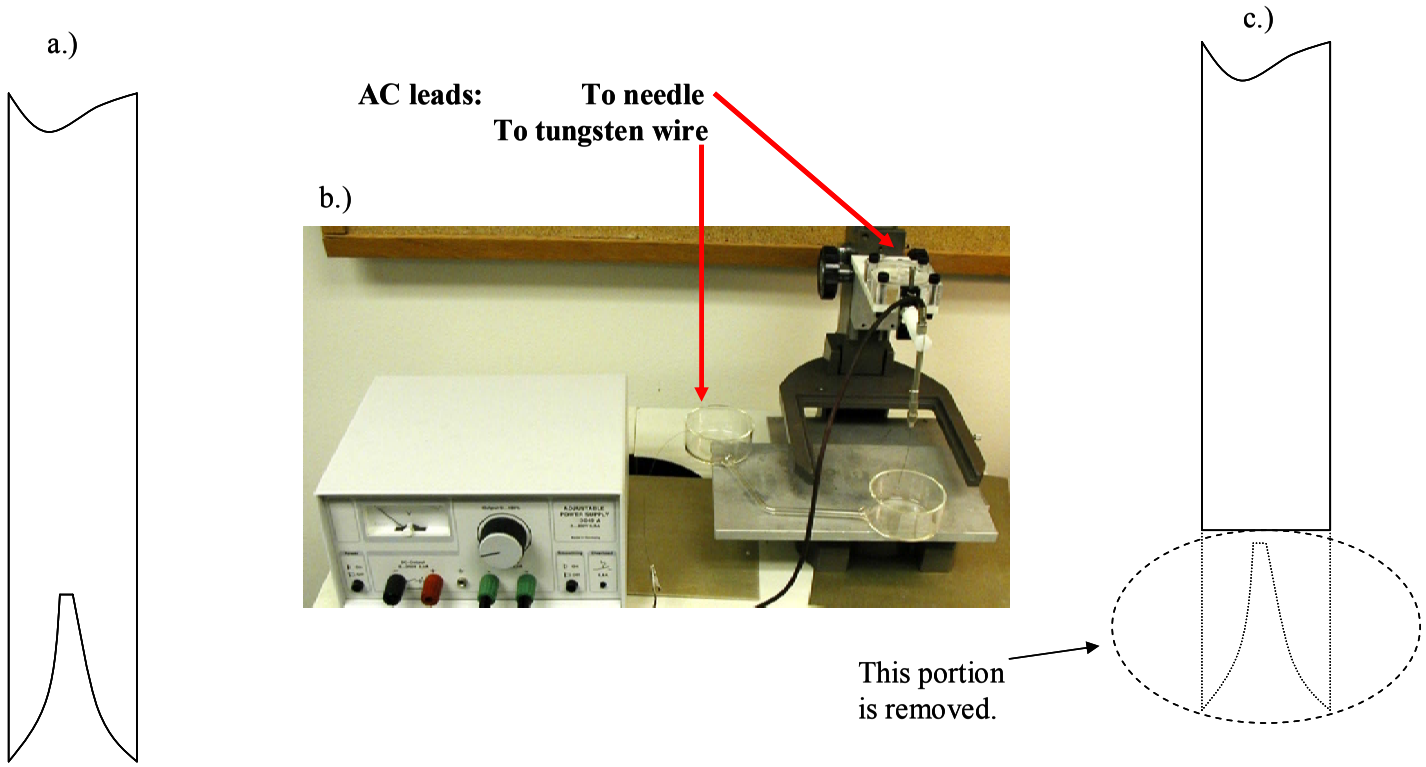


Fig. A-3: Etching the needle.

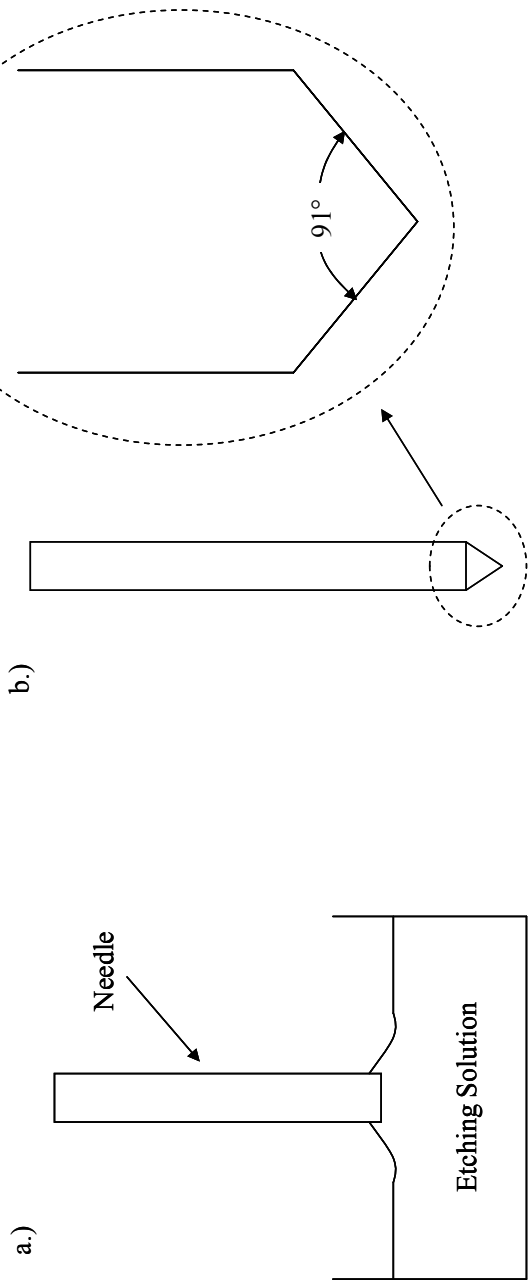
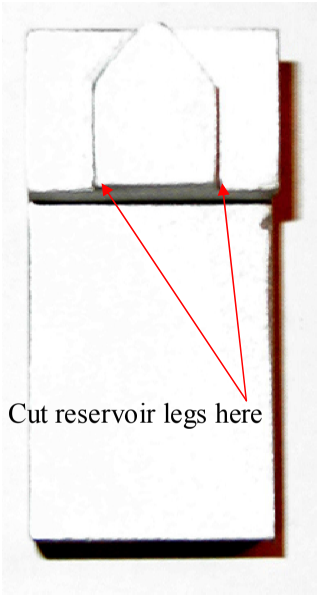


Fig. A-4: Proper placement for etching needle.

a.)



b.)

c.)

Fig. A-5: Placement of reservoir in source assembly.

6.) Attach the electrical connections for etching the reservoir, Fig. A-6, and immerse spring into the etching solution. Apply about 5V AC for 5 minutes to clean. Rinse the spring with distilled water, when finished, to remove any remaining etching solution.

7.) Inserting needle into the source assembly:

   a. Accurately measure 20 mm from tip of needle, cut off excess.

   b. Insert the needle into the source assemble and, using tweezers, position in the middle of the spring reservoir.

   c. It is not necessary that needle be centered in the reservoir but it must be in a vertical position above the spring at a right angle to the top of the spring and 1.3 mm above the top of the spring, Fig. A-7.

8.) Adjust the source assembly to the vertical translator and attach wires to feed-through, Fig. A-8. Place in vacuum chamber and evacuate to at least $1x10^{-6}$ torr. Place a pellet of the Au/Si eutectic (97%Au/3%Si, Academy Precision Metals) in the tantalum boat in the vacuum chamber. Pellet dimensions are a cylinder of 3/8" h, ½ d.

9.) Begin to heat the eutectic by ramping the temperature, of the tantalum boat (monitored with a thermocouple), at a rate of 10° C per minute. The eutectic will melt at 363° C. Once completely melted, check by gently nudging the chamber, you will see ripples in the eutectic if it has melted, maintain the temperature.

10.)The needle and spring must be free of any contaminants before immersing in the eutectic. To do this:

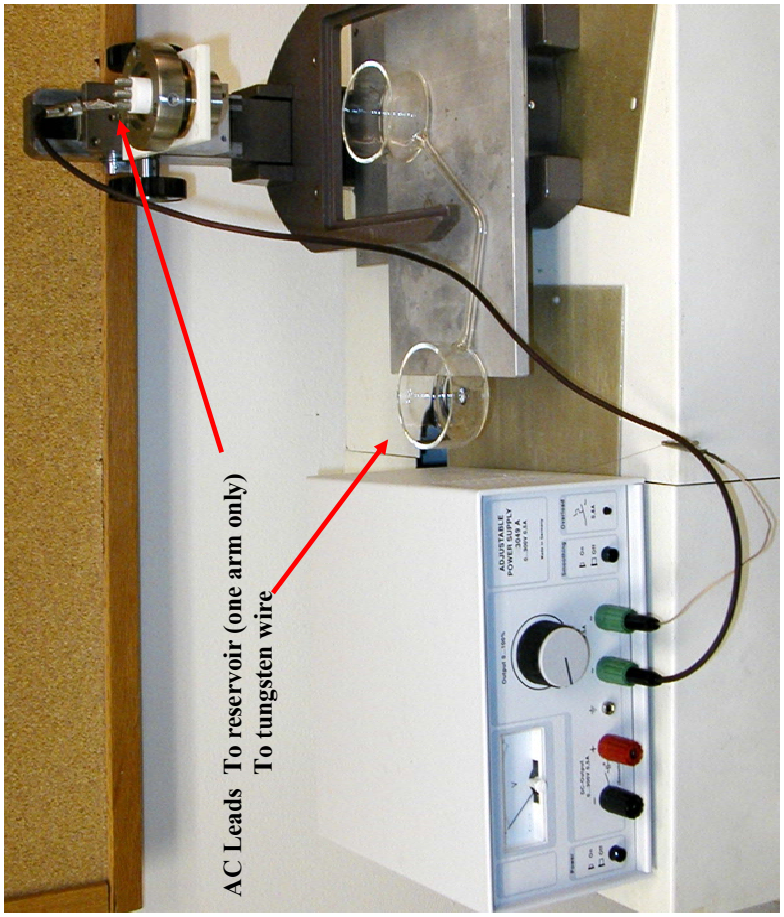


Fig. A-6: Electrical connections for etching reservoir.

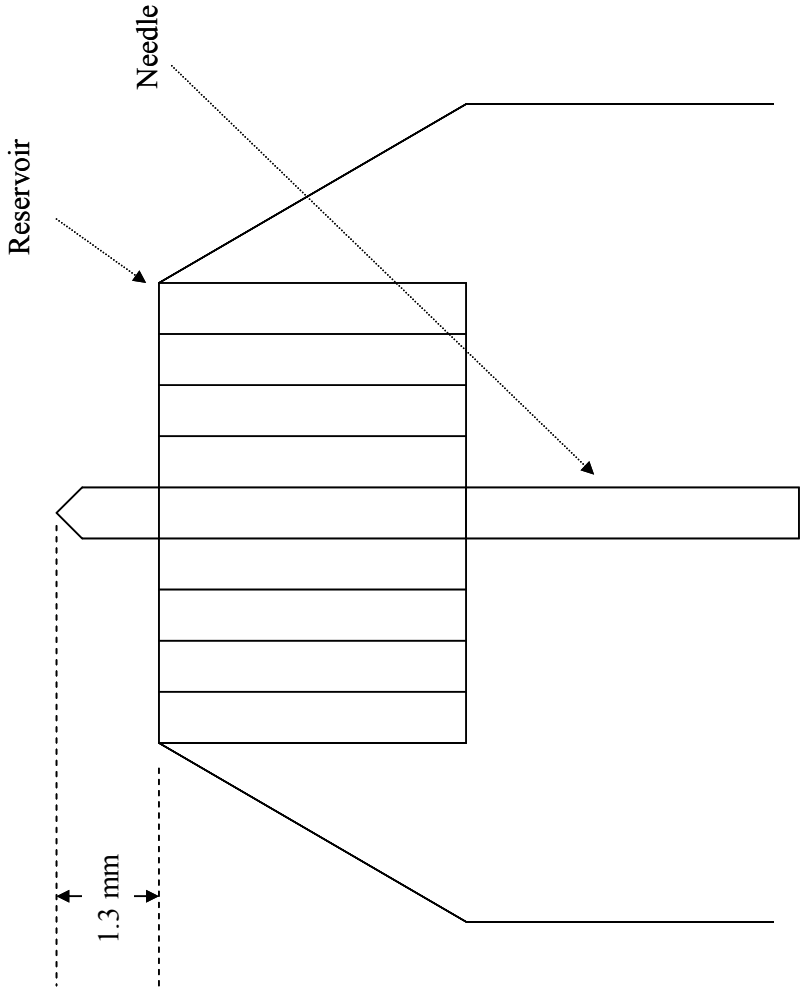


Fig. A-7: Proper placement of needle in reservoir.

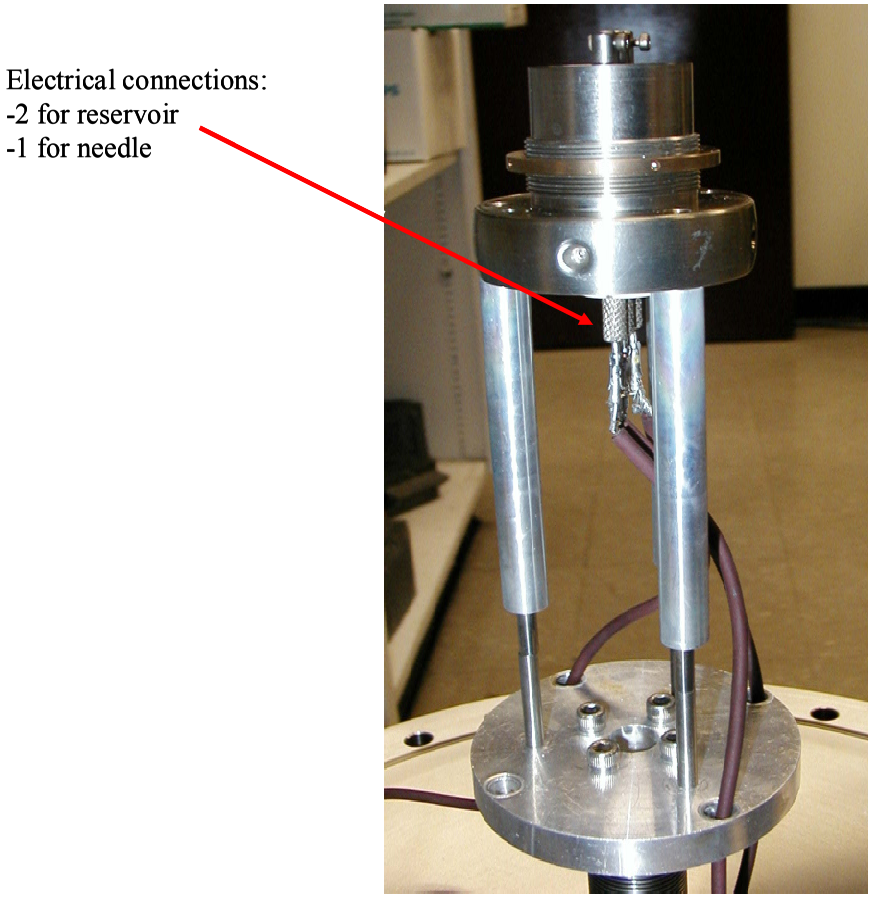


Fig. A-8: Placement of source assembly on vertical translator.

a. Apply 7-8 W between needle and one spring end for about 2 min. Spring and needle will glow brightly, Fig. A-9a.

b. Apply 7-8 W between needle and other end of spring for the same amount of time, Fig. A-9b.

c. Note location of needle and end of spring with divisions on microscope before and during heating. The spring will expand about 10%. And make sure needle does not become detached from spring.

d. Apply about 5 W of power between the spring ends (not the needle), Fig. A-9c, the spring will glow brightly. Lower the needle and spring assembly slowly into the eutectic until the top of the spring is immersed completely. Turn off power to spring and begin to raise it from the melt at a slow but constant rate (not stopping or using jerky motions).

e. Before breaking opening the chamber, examine the needle and reservoir with the microscope. A nice meniscus, of the eutectic, Fig. A-10, should be present between the needle and top of reservoir and the reservoir should be filled with the mixture. If not, repeat this procedure.

11.) Cool the eutectic in the tantalum boat, slowly, to room temperature. Once cooled, open the chamber and remove the source assembly.

12.) Attach the extractor (without diaphragm), Fig. A-11a, and centre the needle using the 4 set screws located at the base, Fig. A-11b. Install diaphragm, in the extractor and make final adjustments to centre the needle (use divisions on microscope). Rotate extractor to bring the needle tip in the plane of the

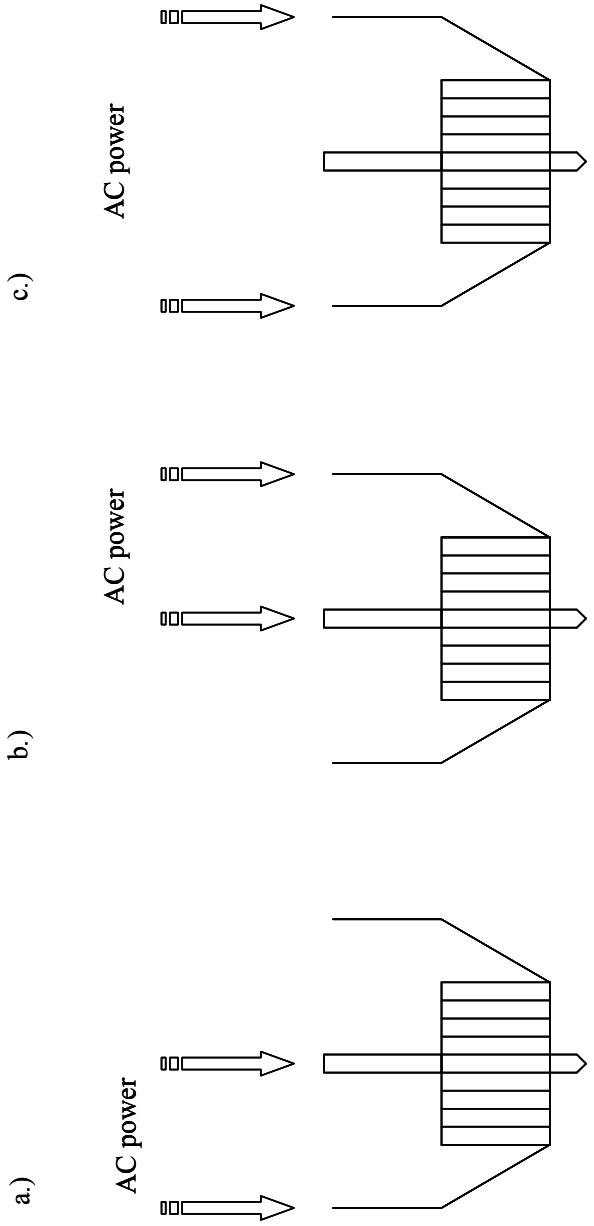


Fig. A-9: Electrical connections for cleaning needle.

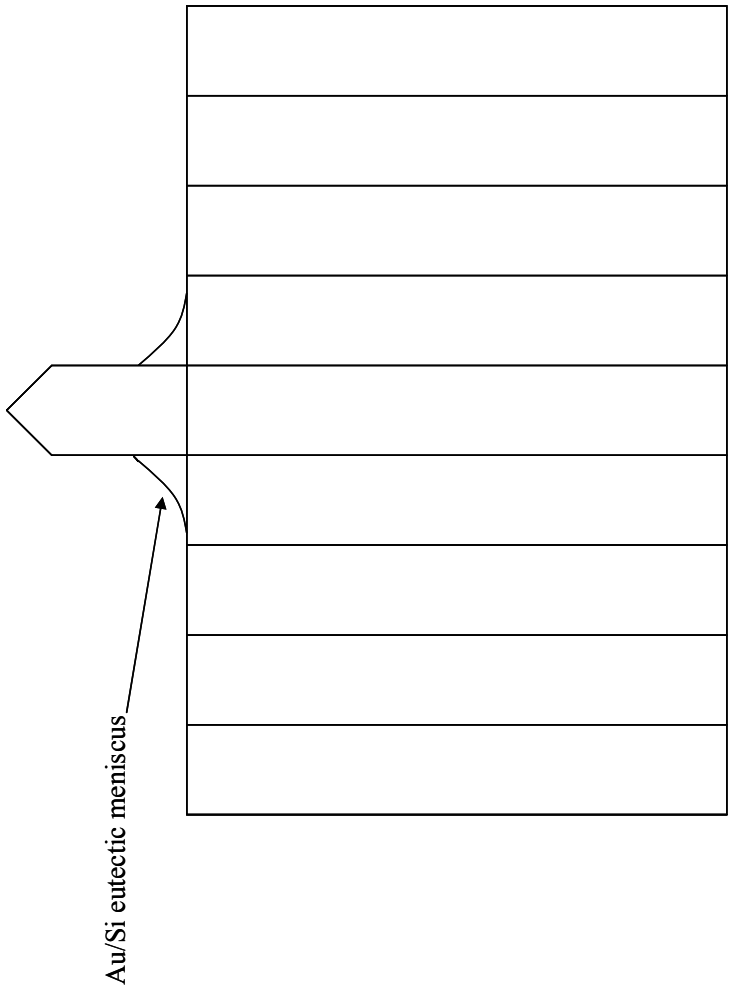


Fig. A-10: Eutectic meniscus from reservoir to needle.

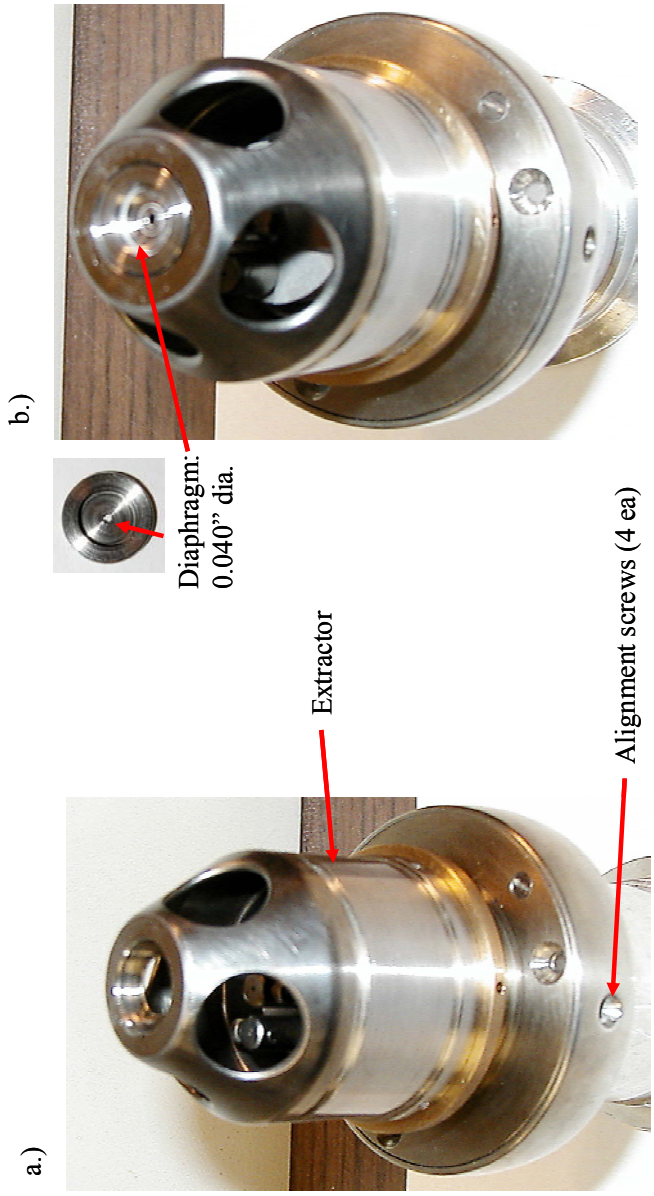


Fig. A-11: Installation and alignment of extractor and diaphragm.

bottom of the diaphragm, again using divisions on microscope. Tighten extractor using set screws.

13.) Finish assembly of source according to Fig. A-12.

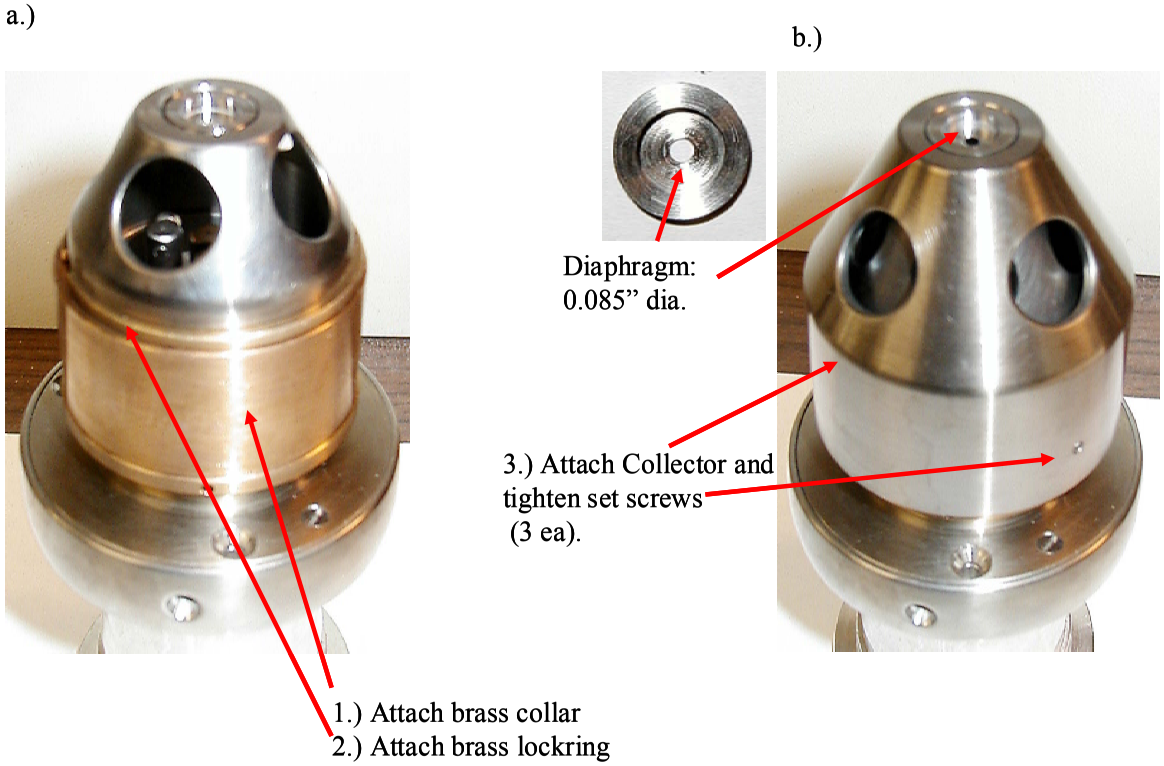


Fig. A-12: Final assembly of the source.

APPENDIX B

CTN-M3 – PCIDIO32-HS CONNECTION

The following is required only for the CTN-M3. The CTN-M4 can be directly interfaced to the data acquisition card in the PC.

The 25 pin D subminiature connector on the CTN-M3 cannot be directly connected to the PCIDIO32 HS data acquisition card in the PC. An interface must be constructed using a CB 68LP connector block (National Instruments, 777145-01). The 25 pin cable is attached to the CTN-M3. Individual conductors, from the cable connected to the CTN-M3, are separated and attached to the connector block according to Fig. B-1. A jumper cable is then installed, connecting terminal 1 on the CB 68LP, to terminals 38 and 40, also on the CB 68LP. Once all of the conductors from the CTN-M3 cable are attached, and the jumpers installed, a R6868 68 pin ribbon cable (National Instruments, 182482-01) is used to connect the PCDIO32-HS data acquisition card to the connector block.

CB 68LP	CTN-M3
10	1
45	2
13	3
48	4
16	5
51	6
53	7
21	8
23	9
58	10
2	11
42	12
3	13

CTN-M3	CB 68LP
14	44
15	12
16	47
17	15
18	17
19	52
20	54
21	22
22	57
23	14
24	49
25	50

Fig. B-1: CTN-M3 / CB 68LP interface.

APPENDIX C

THE TOTAL MATRIX OF EVENTS PROGRAM

Shown below is the code, written in "C" which is used for data acquisition and analysis from the CTN-M3 and CTN-M4 Time-to-Digital Converters.

```
#include <dataacq.h>
#include <utility.h>
#include <ansi_c.h>
#include <analysis.h>
#include <formatio.h>
#include <cvirte.h>
#include <userint.h>
#include "PracticeTotalEvent2.h"
#include "nidaqex.h"
#include <easyio.h>

#define B 2600
#define C 100000
#define D 500000
#define E 32000
#define F 1000
#define G 260

static int mnPanel, expPnl,  menuHandle, multPanel,masscalibPnl, cnspecPnl, normPanel, copy,
                        WriteFileHandle,ReadFileHandle;
char
proj_dir[MAX_PATHNAME_LEN],file_name[MAX_PATHNAME_LEN],file_Name[MAX_PATHNA
ME_LEN];
char File_Name[MAX_PATHNAME_LEN],FILE_name[MAX_PATHNAME_LEN];
char WriteFileName[G], FileNameToWrite[G], PathName[G], name[G], prj_dir[260];
char Date[G], Time[G],Date2[G], Time2[G], operator[G], id[G], target[G], pidat[G], sidat[G],
    startdat[G], stop1dat[G],stop2dat[G], test_name[G],buf[B];
unsigned int size, chanmax;
unsigned int events, ch0, ch1, ch2, ch3, ch4, ch5, ch6, ch7= 0;
unsigned int totalEvents, nullEvents, bigEvents, lochan = 0;
unsigned int stop1Events, stop2Events, stop3Events, stop4Events, stop5Events = 0;
unsigned int stop6Events, stop7Events, stop8Events, stop9Events, stop10Events = 0;
unsigned int stop11Events, stop12Events, stop13Events, stop14Events, stop15Events = 0;
unsigned int stop16Events, stop17Events, stop18Events, stop19Events, stop20Events = 0;
unsigned int stop21Events, stop22Events, stop23Events, stop24Events, stop25Events = 0;
unsigned int stop26Events, stop27Events, stop28Events, stop29Events, stop30Events = 0;
unsigned int stop31Events, stop32Events, stop33Events, stop34Events, stop35Events = 0;
unsigned int stop36Events, stop37Events, stop38Events, stop39Events, stop40Events = 0;
unsigned int stop41Events, stop42Events, stop43Events, stop44Events, stop45Events = 0;
unsigned int stop46Events, stop47Events, stop48Events, stop49Events, stop50Events = 0;
double x1ref, y1ref, x2ref, y2ref, x1, y1, x2, y2, dd, ee = 0;
```

```
double ma, sm, slope, intercept, mse, ca = 0;
int mon, dat, yr, hr, mn, sc, r, Status;
int hit1, hit2, hit3, hit4, hit5, hit6, hit7, hit8, hit9, hit10 = 0;
int hit11, hit12, hit13, hit14, hit15, hit16, hit17, hit18, hit19, hit20 = 0;
int hit21, hit22, hit23, hit24, hit25, hit26, hit27, hit28, hit29, hit30 = 0;
int hit31, hit32, hit33, hit34, hit35, hit36, hit37, hit38, hit39, hit40 = 0;
int hit41, hit42, hit43, hit44, hit45, hit46, hit47, hit48, hit49, hit50 = 0;
int ea1, ea2, ea3, ea4, ea5, ea6, ea7, ea8, ea9, ea10, ea11, ea12, ea13, ea14, ea15, ea16,
   ea17, ea18, ea19, ea20 = 0;
int ea21, ea22, ea23, ea24, ea25, ea26, ea27, ea28, ea29, ea30, ea31, ea32, ea33, ea34, ea35,
   ea36, ea37, ea38 = 0;
int ea39, ea40,ea41, ea42, ea43, ea44, ea45, ea46, ea47, ea48, ea49, ea50= 0;
int response, response1, response2, pen, en, nHit, n = 0;
int ch0en, ch1en, ch2en, ch3en, ch4en, ch5en, ch6en, ch7en = 0;
int d, z, x, y,q,e, llm, ulm, det, shift, ed = 0;
unsigned short *array;
unsigned int tempD[2000]={0};
unsigned int tempT[2000]={0};
unsigned int
*Event1TArray,*Event1DArray,*Event2TArray,*Event2DArray,*Event3TArray,*Event3DArray;
unsigned int
*Event4TArray,*Event4DArray,*Event5TArray,*Event5DArray,*Event6TArray,*Event6DArray;
unsigned int
*Event7TArray,*Event7DArray,*Event8TArray,*Event8DArray,*Event9TArray,*Event9DArray;
unsigned int
*Event10TArray,*Event10DArray,*Event11TArray,*Event11DArray,*Event12TArray,*Event12DArray;
unsigned int
*Event13TArray,*Event13DArray,*Event14TArray,*Event14DArray,*Event15TArray,*Event15DArray;
unsigned int
*Event16TArray,*Event16DArray,*Event17TArray,*Event17DArray,*Event18TArray,*Event18DArray;
unsigned int
*Event19TArray,*Event19DArray,*Event20TArray,*Event20DArray,*Event21TArray,*Event21DArray;
unsigned int
*Event22TArray,*Event22DArray,*Event23TArray,*Event23DArray,*Event24TArray,*Event24DArray;
unsigned int
*Event25TArray,*Event25DArray,*Event26TArray,*Event26DArray,*Event27TArray,*Event27DArray;
unsigned int
*Event28TArray,*Event28DArray,*Event29TArray,*Event29DArray,*Event30TArray,*Event30DArray;
unsigned int
*Event31TArray,*Event31DArray,*Event32TArray,*Event32DArray,*Event33TArray,*Event33DArray;
unsigned int
*Event34TArray,*Event34DArray,*Event35TArray,*Event35DArray,*Event36TArray,*Event36DArray;
unsigned int
*Event37TArray,*Event37DArray,*Event38TArray,*Event38DArray,*Event39TArray,*Event39DArray;
unsigned int
*Event40TArray,*Event40DArray,*Event41TArray,*Event41DArray,*Event42TArray,*Event42DArray;
unsigned int
*Event43TArray,*Event43DArray,*Event44TArray,*Event44DArray,*Event45TArray,*Event45DArray;
unsigned int
*Event46TArray,*Event46DArray,*Event47TArray,*Event47DArray,*Event48TArray,*Event48DArray;
unsigned int *Event49TArray,*Event49DArray,*Event50TArray,*Event50DArray;
unsigned int *coinspec,*currTArray,*currDArray;
 int *coinspec0,*coinspec1,*coinspec2,*coinspec3;
```

```
 int *coinspec4,*coinspec5,*coinspec6,*coinspec7;
unsigned int MultArray[51] = {0};
unsigned int MultXArray[51] = {0};
unsigned int *currHist;
double      *normArray;
double      *Mass;
double MassCalibY[8],MassCalibX[8],MassCalib[8] = {0};
short statai;
short iStatus = 0;
short iRetVal = 0;
short iDevice = 1;
short iGroup = 1;
short iPort = 0;
short iDir = 0;
short iSignal = 1;
short iEdge = 0;
short iAckDelayTime =1;
short iDBModeON = 1;
short iDBModeOFF = 0;
short iOldDataStop = 1;
short iPartialTransfer = 0;
short iHalfReady = 0;
unsigned long iLoopCount = 0;
unsigned long iHalfBufsToRead = C;
unsigned long ulAlignIndex = 0;
short iResource = 11;
short iIgnoreWarning = 1;
unsigned long lTimeout = 180;
short iYieldON = 1;

short iGroupSize = 2;
short iReqPol = 1;
short iAckPol = 1;
unsigned short *piBuffer;
unsigned short *piHalfBuffer;
unsigned long ulCount = E;
unsigned long ulPtsTfr = (E/2);
unsigned long ulBufferSize = E;

/*****************************************************Main and
Exit**********************/
int main (int argc, char *argv[])
{
        if (InitCVIRTE (0, argv, 0) == 0)
                return -1;          /* out of memory */
        if ((mnPanel = LoadPanel (0, "PracticeTotalEvent2.uir", MN_PANEL)) < 0)
                return -1;

        menuHandle = LoadMenuBar(mnPanel,"PracticeTotalEvent2.uir", MAIN_MENU);
        multPanel = LoadPanel (mnPanel, "PracticeTotalEvent2.uir", MULTPNL);
        masscalibPnl = LoadPanel (mnPanel, "PracticeTotalEvent2.uir", CALIBPNL);
        cnspecPnl = LoadPanel (mnPanel, "PracticeTotalEvent2.uir", COINPNL);
        normPanel = LoadPanel (mnPanel, "PracticeTotalEvent2.uir", NORMPNL);
```

```
	expPnl = LoadPanel (mnPanel,"PracticeTotalEvent2.uir", EXP_PNL);
	DisplayPanel (mnPanel);
	RunUserInterface ();
	DiscardPanel (mnPanel);

	return 0;
}
int CVICALLBACK ExitCB (int panel, int control, int event,
		void *callbackData, int eventData1, int eventData2)
{
	switch (event)
		{
		case EVENT_COMMIT:
			QuitUserInterface (0);
			break;
		}
	return 0;
}
void CVICALLBACK MenuExitCB (int menuBar, int menuItem, void *callbackData,
		int panel)
{
			QuitUserInterface (0);
}
/*************************Array Allocation
Functions***********************************/
void histpiBuffer(int maxChannel){
	piBuffer=( unsigned short *)calloc(maxChannel, sizeof (unsigned short));
                                                                        }
void histpiHalfBuffer(int maxChannel){
	piHalfBuffer=( unsigned short *)calloc(maxChannel, sizeof (unsigned short));
                                                                            }
void histArray(int maxChannel){
	array=(unsigned short *)calloc(maxChannel+1, sizeof (unsigned short));
                                                                        }
void histCoinspec(int maxChannel){
	coinspec=(int *)calloc(maxChannel, sizeof (int));
                                                                            }
void histAllocate0(int maxChannel){
	coinspec0=(int *)calloc(maxChannel+1, sizeof (int));
                                                                            }
void histAllocate1(int maxChannel){

	coinspec1=(int *)calloc(maxChannel+1, sizeof (int));
                                                                        }
void histAllocate2(int maxChannel){
	coinspec2=(int *)calloc(maxChannel+1, sizeof (int));
                                                                            }
void histAllocate3(int maxChannel){

	coinspec3=(int *)calloc(maxChannel+1, sizeof (int));
                                                                        }
void histAllocate4(int maxChannel){
	coinspec4=(int *)calloc(maxChannel+1, sizeof (int));
```

```
                                                                    }
void histAllocate5(int maxChannel){

        coinspec5=(int *)calloc(maxChannel+1, sizeof (int));
                                                                    }
void histAllocate6(int maxChannel){
        coinspec6=(int *)calloc(maxChannel+1, sizeof (int));
                                                                    }
void histAllocate7(int maxChannel){

        coinspec7=(int *)calloc(maxChannel+1, sizeof (int));
                                                                    }
void histAllocateEA1(int maxChannel){

        Event1TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event1DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                    }
void histAllocateEA2(int maxChannel){

        Event2TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event2DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                    }
void histAllocateEA3(int maxChannel){

        Event3TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event3DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                    }
void histAllocateEA4(int maxChannel){

        Event4TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event4DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                    }
void histAllocateEA5(int maxChannel){

        Event5TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event5DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                    }
void histAllocateEA6(int maxChannel){

        Event6TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event6DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                    }
void histAllocateEA7(int maxChannel){

        Event7TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event7DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                    }
void histAllocateEA8(int maxChannel){

        Event8TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event8DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                    }
```

```
void histAllocateEA9(int maxChannel){

        Event9TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event9DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA10(int maxChannel){

        Event10TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event10DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA11(int maxChannel){

        Event11TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event11DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA12(int maxChannel){

        Event12TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event12DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA13(int maxChannel){

        Event13TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event13DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA14(int maxChannel){

        Event14TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event14DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA15(int maxChannel){

        Event15TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event15DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA16(int maxChannel){

        Event16TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event16DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA17(int maxChannel){

        Event17TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event17DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA18(int maxChannel){

        Event18TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event18DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA19(int maxChannel){

        Event19TArray=(int *)calloc(maxChannel+1, sizeof (int));
```

```
        Event19DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA20(int maxChannel){

        Event20TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event20DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA21(int maxChannel){

        Event21TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event21DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA22(int maxChannel){

        Event22TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event22DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA23(int maxChannel){

        Event23TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event23DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA24(int maxChannel){

        Event24TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event24DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA25(int maxChannel){

        Event25TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event25DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA26(int maxChannel){

        Event26TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event26DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA27(int maxChannel){

        Event27TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event27DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA28(int maxChannel){

        Event28TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event28DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA29(int maxChannel){

        Event29TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event29DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
```

```
void histAllocateEA30(int maxChannel){

        Event30TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event30DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA31(int maxChannel){

        Event31TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event31DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA32(int maxChannel){

        Event32TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event32DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA33(int maxChannel){

        Event33TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event33DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA34(int maxChannel){

        Event34TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event34DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA35(int maxChannel){

        Event35TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event35DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA36(int maxChannel){

        Event36TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event36DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA37(int maxChannel){

        Event37TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event37DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA38(int maxChannel){

        Event38TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event38DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA39(int maxChannel){

        Event39TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event39DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA40(int maxChannel){

        Event40TArray=(int *)calloc(maxChannel+1, sizeof (int));
```

```
        Event40DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA41(int maxChannel){

        Event41TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event41DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA42(int maxChannel){

        Event42TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event42DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA43(int maxChannel){

        Event43TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event43DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA44(int maxChannel){

        Event44TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event44DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA45(int maxChannel){

        Event45TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event45DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA46(int maxChannel){

        Event46TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event46DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA47(int maxChannel){

        Event47TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event47DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA48(int maxChannel){

        Event48TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event48DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA49(int maxChannel){

        Event49TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event49DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
void histAllocateEA50(int maxChannel){

        Event50TArray=(int *)calloc(maxChannel+1, sizeof (int));
        Event50DArray=(int *)calloc(maxChannel+1, sizeof (int));
                                                                }
```

```
void MassArray(int maxChannel){

        Mass=(double *)calloc(maxChannel+1, sizeof (double));
                                                                }
void normArr(int maxChannel){

        normArray=(double *)calloc(maxChannel+1, sizeof (double));
                                                                }
/*****************************************Program 6 CTN-
M3*********************************************************/
void StartDataAcquire6CB ()
{
if (FileSelectPopup (proj_dir, "*.cns", "*.cns", "Name of File to Save",
                                        VAL_SAVE_BUTTON, 0, 1, 1, 1, file_Name) > 0){

        int hi, x, z = 0;
        iGroupSize = 2;
        iReqPol = 1;
        iAckPol = 1;
        ulCount = F;
        ulPtsTfr = (F/2);
        ulBufferSize = F;

        GetCtrlVal (expPnl, EXP_PNL_MAXCHAN, &chanmax);
        SetCtrlVal (expPnl, EXP_PNL_TM1TXTBOX, TimeStr());
        SetCtrlVal (expPnl, EXP_PNL_DT1TXTBOX, DateStr());
        histpiBuffer(F);
        histpiHalfBuffer(F/2);
        histCoinspec((chanmax+1000));

        for(z=0;z<=F-1;z++){

                piBuffer[z]=0;

                                        }//for


   iStatus = Timeout_Config(iDevice, lTimeout);

   iRetVal = NIDAQErrorHandler(iStatus, "Timeout_Config",
    iIgnoreWarning);

   iStatus = DIG_Grp_Config(iDevice, iGroup, iGroupSize, iPort,
    iDir);

   iRetVal = NIDAQErrorHandler(iStatus, "DIG_Grp_Config",
    iIgnoreWarning);

   iStatus = DIG_Grp_Mode(iDevice, iGroup, iSignal, iEdge, iReqPol,
    iAckPol, iAckDelayTime);

   iRetVal = NIDAQErrorHandler(iStatus, "DIG_Grp_Mode",
    iIgnoreWarning);
```

```
iStatus = DIG_DB_Config(iDevice, iGroup, iDBModeON, iOldDataStop,
 iPartialTransfer);

iRetVal = NIDAQErrorHandler(iStatus, "DIG_DB_Config",
 iIgnoreWarning);

iStatus = Align_DMA_Buffer(iDevice, iResource, piBuffer, ulCount,
 ulBufferSize, &ulAlignIndex);

iRetVal = NIDAQErrorHandler(iStatus, "Align_DMA_Buffer",
 iIgnoreWarning);

iStatus = DIG_Block_In(iDevice, iGroup, piBuffer, ulCount);

iRetVal = NIDAQErrorHandler(iStatus, "DIG_Block_In",
 iIgnoreWarning);

        ++iLoopCount;
while ((iLoopCount<iHalfBufsToRead) && (iStatus == 0)) {

   iStatus = DIG_DB_HalfReady(iDevice, iGroup, &iHalfReady);

   if (iStatus >= 0) {

      if (iHalfReady == 1) {

         iStatus = DIG_DB_Transfer(iDevice, iGroup,
          piHalfBuffer, ulPtsTfr);

         iRetVal = NIDAQErrorHandler(iStatus,
          "DIG_DB_Transfer", iIgnoreWarning);

                for(x=0;x<=(F/4)-1;x++){

                                                hi = 0;

                                                hi=piHalfBuffer[x];      // Converts buffer elements
from negative to positive logic

                                                                        //
                                        hi^=65535;
                                                //

                                                                        //
                                        piHalfBuffer[x] = hi;
                                        if (hi<chanmax){
                                        coinspec[hi]+=1;                        //
                                                                }
                                                        } // for
```

```
                                        DeleteGraphPlot (mnPanel, MN_PANEL_GRAPH, -1,
VAL_IMMEDIATE_DRAW);
                                        RefreshGraph (mnPanel, MN_PANEL_GRAPH);
                                        PlotY (mnPanel, MN_PANEL_GRAPH, coinspec, chanmax+1000,
VAL_INTEGER,
                                                VAL_THIN_LINE, VAL_EMPTY_SQUARE,
VAL_SOLID, 1, VAL_BLACK);
                        ProcessSystemEvents();


                        }// if status > 0

                        }// if halfready == 1

        else {

           iRetVal = NIDAQErrorHandler(iStatus, "DIG_DB_HalfReady",
            iIgnoreWarning);

           }// else

        iRetVal = NIDAQYield(iYieldON);

           }//while

        iStatus = DIG_Block_Clear(iDevice, iGroup);

        iStatus = DIG_DB_Config(iDevice, iGroup, iDBModeOFF, iOldDataStop,
                        iPartialTransfer);

        iStatus = DIG_Grp_Config(iDevice, iGroup, 0, 0, 0);

        iStatus = Timeout_Config(iDevice, -1);
}
        MessagePopup ("CTN-M3", "Data Acquisition Complete");
                        ArrayToFile (file_Name, coinspec, VAL_INTEGER, (chanmax + 1000), 1,
                                        VAL_GROUPS_TOGETHER,
VAL_GROUPS_AS_COLUMNS,
                                        VAL_CONST_WIDTH, 10, VAL_BINARY,
VAL_APPEND);

}//void start prog 6
/*****************************************Program 8 CTN-
M3*********************************************************/
void StartDataAcquire8CB ()
{

if (FileSelectPopup (proj_dir, "*.CM3", "*.CM3", "Name of File to Save",
                                        VAL_SAVE_BUTTON, 0, 1, 1, 1, File_Name) > 0){

            int x, hi = 0;
            iGroupSize = 2;
            iReqPol = 1;
```

```
        iAckPol = 1;
        ulCount = E;
        ulPtsTfr = (E/2);
        ulBufferSize = E;
        SetCtrlVal (expPnl, EXP_PNL_TM1TXTBOX, TimeStr());
        SetCtrlVal (expPnl, EXP_PNL_DT1TXTBOX, DateStr());
        histpiBuffer(E);
        histpiHalfBuffer(E/2);

        for(z=0;z<=E-1;z++){

                piBuffer[z]=0;

                                                }//for


iStatus = Timeout_Config(iDevice, lTimeout);

iRetVal = NIDAQErrorHandler(iStatus, "Timeout_Config",
 iIgnoreWarning);

iStatus = DIG_Grp_Config(iDevice, iGroup, iGroupSize, iPort,
 iDir);

iRetVal = NIDAQErrorHandler(iStatus, "DIG_Grp_Config",
 iIgnoreWarning);

iStatus = DIG_Grp_Mode(iDevice, iGroup, iSignal, iEdge, iReqPol,
 iAckPol, iAckDelayTime);

iRetVal = NIDAQErrorHandler(iStatus, "DIG_Grp_Mode",
 iIgnoreWarning);

iStatus = DIG_DB_Config(iDevice, iGroup, iDBModeON, iOldDataStop,
 iPartialTransfer);

iRetVal = NIDAQErrorHandler(iStatus, "DIG_DB_Config",
 iIgnoreWarning);

iStatus = Align_DMA_Buffer(iDevice, iResource, piBuffer, ulCount,
 ulBufferSize, &ulAlignIndex);

iRetVal = NIDAQErrorHandler(iStatus, "Align_DMA_Buffer",
 iIgnoreWarning);

iStatus = DIG_Block_In(iDevice, iGroup, piBuffer, ulCount);

iRetVal = NIDAQErrorHandler(iStatus, "DIG_Block_In",
 iIgnoreWarning);

++iLoopCount;

while ((iLoopCount<iHalfBufsToRead) && (iStatus == 0)) {
```

```
iStatus = DIG_DB_HalfReady(iDevice, iGroup, &iHalfReady);

if (iStatus >= 0) {

   if (iHalfReady == 1) {

      iStatus = DIG_DB_Transfer(iDevice, iGroup,
       piHalfBuffer, ulPtsTfr);

      iRetVal = NIDAQErrorHandler(iStatus,
       "DIG_DB_Transfer", iIgnoreWarning);

            for(x=0;x<=(E/2)-1;x++){

                                          hi = 0;

                                          hi=piHalfBuffer[x];       // Converts buffer elements
from negative to positive logic

                                                                    //
                                    hi^=65535;
                                                //

                                                                    //
                                    piHalfBuffer[x] = hi;
                                //

                                                                    //
                                    if ((piHalfBuffer[x]&64512) == 51200) {

                                                                    events+=1;


                              }

                                                                                  } // for

      ProcessSystemEvents();

                                    SetCtrlVal (mnPanel, MN_PANEL_NUMEVENT, events);
                              ArrayToFile (File_Name, piHalfBuffer,
VAL_UNSIGNED_SHORT_INTEGER,
                                                                    E/2, 1, VAL_GROUPS_TOGETHER,
VAL_GROUPS_AS_COLUMNS,
                                                                    VAL_CONST_WIDTH, 10,
VAL_BINARY, VAL_APPEND);


            }}

   else {
```

```
            iRetVal = NIDAQErrorHandler(iStatus, "DIG_DB_HalfReady",
             iIgnoreWarning);

             }

          iRetVal = NIDAQYield(iYieldON);

      }

      iStatus = DIG_Block_Clear(iDevice, iGroup);

      iStatus = DIG_DB_Config(iDevice, iGroup, iDBModeOFF, iOldDataStop,
       iPartialTransfer);

      iStatus = DIG_Grp_Config(iDevice, iGroup, 0, 0, 0);

      iStatus = Timeout_Config(iDevice, -1);
}
             MessagePopup ("CTN-M3", "Data Acquisition Complete");

}
/***********************************************Program A CTN-
M4*****************************************************/
void StartDataAcquireA()
{

             if (FileSelectPopup (proj_dir, "*.CM4", "*.CM4", "Name of File to Save",
                                                          VAL_SAVE_BUTTON, 0, 1, 1, 1, FILE_name) >
0){

             unsigned short z = 0;
             iSignal = 0;
             iGroupSize = 4;
             iReqPol = 0;
             iAckPol = 0;
             ulCount = (E/4);
             ulPtsTfr = (E/4);
             ulBufferSize = E;
             SetCtrlVal (expPnl, EXP_PNL_TM1TXTBOX, TimeStr());
             SetCtrlVal (expPnl, EXP_PNL_DT1TXTBOX, DateStr());
             histpiBuffer(E);
             histpiHalfBuffer(E/4);

      iStatus = Timeout_Config(iDevice, lTimeout);

      iRetVal = NIDAQErrorHandler(iStatus, "Timeout_Config",
       iIgnoreWarning);

      iStatus = DIG_Grp_Config(iDevice, iGroup, iGroupSize, iPort,
       iDir);

      iRetVal = NIDAQErrorHandler(iStatus, "DIG_Grp_Config",
       iIgnoreWarning);
```

```
iStatus = DIG_Grp_Mode(iDevice, iGroup, iSignal, iEdge, iReqPol,
 iAckPol, iAckDelayTime);

iRetVal = NIDAQErrorHandler(iStatus, "DIG_Grp_Mode",
 iIgnoreWarning);

iStatus = DIG_DB_Config(iDevice, iGroup, iDBModeON, iOldDataStop,
 iPartialTransfer);

iRetVal = NIDAQErrorHandler(iStatus, "DIG_DB_Config",
 iIgnoreWarning);

iStatus = Align_DMA_Buffer(iDevice, iResource, piBuffer, ulCount,
 ulBufferSize, &ulAlignIndex);

iRetVal = NIDAQErrorHandler(iStatus, "Align_DMA_Buffer",
 iIgnoreWarning);

iStatus = DIG_Block_In(iDevice, iGroup, piBuffer, ulCount);

iRetVal = NIDAQErrorHandler(iStatus, "DIG_Block_In",
 iIgnoreWarning);

        WriteToDigitalLine (1, "4", 1, 4, 1, 1);
        WriteToDigitalLine (1, "4", 1, 4, 1, 0);
        WriteToDigitalLine (1, "4", 3, 4, 1, 1);

while ((iLoopCount < iHalfBufsToRead) && (iStatus == 0)) {

   iStatus = DIG_DB_HalfReady(iDevice, iGroup, &iHalfReady);

   if (iStatus >= 0) {

     if (iHalfReady == 1) {

        iStatus = DIG_DB_Transfer(iDevice, iGroup,
         piHalfBuffer, ulPtsTfr);

        iRetVal = NIDAQErrorHandler(iStatus,
         "DIG_DB_Transfer", iIgnoreWarning);

        ++iLoopCount;

        for(z=0;z<=(E/4)-1;z++){

        if((piHalfBuffer[z]&61440) == 32768){

                                        events+=1;
                                                                }
                                                }
```

```
                                        SetCtrlVal (mnPanel, MN_PANEL_NUMEVENT, events);

                                        ArrayToFile (FILE_name, piHalfBuffer,
VAL_UNSIGNED_SHORT_INTEGER, E/4, 1,
                                                                VAL_GROUPS_TOGETHER,
VAL_GROUPS_AS_COLUMNS,
                                                                VAL_CONST_WIDTH, 10,
VAL_BINARY, VAL_APPEND);

                                                                ProcessSystemEvents();
            }


        }
        else {

            iRetVal = NIDAQErrorHandler(iStatus, "DIG_DB_HalfReady",
             iIgnoreWarning);

              }//else

        iRetVal = NIDAQYield(iYieldON);

    }

               WriteToDigitalLine (1, "4", 3, 4, 1, 0);

    iStatus = DIG_Block_Clear(iDevice, iGroup);

    iStatus = DIG_DB_Config(iDevice, iGroup, iDBModeOFF, iOldDataStop,
     iPartialTransfer);

    iStatus = DIG_Grp_Config(iDevice, iGroup, 0, 0, 0);

    iStatus = Timeout_Config(iDevice, -1);

}

            MessagePopup ("CTN-M4", "Data Acquisition Complete");
} //void Program A
/****************************************Coincidence
Routine*****************************/

void convertData() {

            int x, y, z, q, w, v = 0;

                                        w=0;
                                        GetCtrlVal (cnspecPnl, COINPNL_LOCHANUM, &lochan);
                                        for(x=0;x<=((nHit*n)-1);x++){

                                                    for(y=0;y<=(n-1);y++)     {
```

```
                                                    tempT[y]=currTArray[x];
                                                    tempD[y]=currDArray[x];
                                                     x++;

                                                                                    }
                                                if(tempT[n-1]>lochan){
                                                for(y=0;y<=(n-1);y++){
                                                                                    if
((tempD[y] == 32768) || (tempD[y] == 0))    {tempD[y] = 0;}
                                                                                    if
((tempD[y] == 33792) || (tempD[y] == 4096)) {tempD[y] = 1;}
                                                                                    if
((tempD[y] == 34816) || (tempD[y] == 8192)) {tempD[y] = 2;}
                                                                                    if
((tempD[y] == 35840) || (tempD[y] == 12288)){tempD[y] = 3;}
                                                                                    if
((tempD[y] == 36864) || (tempD[y] == 16384)){tempD[y] = 4;}
                                                                                    if
((tempD[y] == 37888) || (tempD[y] == 20480)){tempD[y] = 5;}
                                                                                    if
((tempD[y] == 38912) || (tempD[y] == 24576)){tempD[y] = 6;}
                                                                                    if
((tempD[y] == 39936) || (tempD[y] == 28672)){tempD[y] = 7;}

                                                                                }

                                                        for(z=0;z<=(n-1);z++){

                                                                    if((tempT[z]<=ulm) &&
(tempT[z]>=llm)&&tempD[z]==en){

                        for(q=0;q<=(n-1);q++){


                                    det=tempD[q];

                                d=tempT[q];



                            if (d>0){

                        switch (det)

                                {
```

```
                    case 0 : coinspec0[d]+=1;  break;
                    case 1 : coinspec1[d]+=1;  break;
                    case 2 : coinspec2[d]+=1;  break;
                    case 3 : coinspec3[d]+=1;  break;
                    case 4 : coinspec4[d]+=1;  break;
                    case 5 : coinspec5[d]+=1;  break;
                    case 6 : coinspec6[d]+=1;  break;
                    case 7 : coinspec7[d]+=1;  break;
            }//switch
            }//if
            if(response1>0){

                        currTArray[w]=d;
                        currDArray[w]=det;
                        w++;
                            }//if

                                }//for
z=n;
                                 }//if
                                                          }//////////
                                                           }//for
                                x--;
                                        }//for
                    switch (n)
                            {
```

```
                                                    case
1:Event1TArray=currTArray;Event1DArray=currDArray;
                                                    if (response1>0){v=w;SetCtrlVal (multPanel,
MULTPNL_1HITNUM, v);}break;
                                                    case
2:Event2TArray=currTArray;Event2DArray=currDArray;
                                                    if (response1>0){v=w/2;SetCtrlVal (multPanel,
MULTPNL_2HITNUM, v);}break;
                                                    case
3:Event3TArray=currTArray;Event3DArray=currDArray;
                                                    if (response1>0){v=w/3;SetCtrlVal (multPanel,
MULTPNL_3HITNUM, v);}break;
                                                    case
4:Event4TArray=currTArray;Event4DArray=currDArray;
                                                    if (response1>0){v=w/4;SetCtrlVal (multPanel,
MULTPNL_4HITNUM, v);}break;
                                                    case
5:Event5TArray=currTArray;Event5DArray=currDArray;
                                                    if (response1>0){v=w/5;SetCtrlVal (multPanel,
MULTPNL_5HITNUM, v);}break;
                                                    case
6:Event6TArray=currTArray;Event6DArray=currDArray;
                                                    if (response1>0){v=w/6;SetCtrlVal (multPanel,
MULTPNL_6HITNUM, v);}break;
                                                    case
7:Event7TArray=currTArray;Event7DArray=currDArray;
                                                    if (response1>0){v=w/7;SetCtrlVal (multPanel,
MULTPNL_7HITNUM, v);}break;
                                                    case
8:Event8TArray=currTArray;Event8DArray=currDArray;
                                                    if (response1>0){v=w/8;SetCtrlVal (multPanel,
MULTPNL_8HITNUM, v);}break;
                                                    case
9:Event9TArray=currTArray;Event9DArray=currDArray;
                                                    if (response1>0){v=w/9;SetCtrlVal (multPanel,
MULTPNL_9HITNUM, v);}break;
                                                    case
10:Event10TArray=currTArray;Event10DArray=currDArray;
                                                    if (response1>0){v=w/10;SetCtrlVal (multPanel,
MULTPNL_10HITNUM, v);}break;
                                                    case
11:Event11TArray=currTArray;Event11DArray=currDArray;
                                                    if (response1>0){v=w/11;SetCtrlVal (multPanel,
MULTPNL_11HITNUM, v);}break;
                                                    case
12:Event12TArray=currTArray;Event12DArray=currDArray;
                                                    if (response1>0){v=w/12;SetCtrlVal (multPanel,
MULTPNL_12HITNUM, v);}break;
                                                    case
13:Event13TArray=currTArray;Event13DArray=currDArray;
                                                    if (response1>0){v=w/13;SetCtrlVal (multPanel,
MULTPNL_13HITNUM, v);}break;
```

```
                                                    case
14:Event14TArray=currTArray;Event14DArray=currDArray;
                                                    if (response1>0){v=w/14;SetCtrlVal (multPanel,
MULTPNL_14HITNUM, v);}break;
                                                    case
15:Event15TArray=currTArray;Event15DArray=currDArray;
                                                    if (response1>0){v=w/15;SetCtrlVal (multPanel,
MULTPNL_15HITNUM, v);}break;
                                                    case
16:Event16TArray=currTArray;Event16DArray=currDArray;
                                                    if (response1>0){v=w/16;SetCtrlVal (multPanel,
MULTPNL_16HITNUM, v);}break;
                                                    case
17:Event17TArray=currTArray;Event17DArray=currDArray;
                                                    if (response1>0){v=w/17;SetCtrlVal (multPanel,
MULTPNL_17HITNUM, v);}break;
                                                    case
18:Event18TArray=currTArray;Event18DArray=currDArray;
                                                    if (response1>0){v=w/18;SetCtrlVal (multPanel,
MULTPNL_18HITNUM, v);}break;
                                                    case
19:Event19TArray=currTArray;Event19DArray=currDArray;
                                                    if (response1>0){v=w/19;SetCtrlVal (multPanel,
MULTPNL_19HITNUM, v);}break;
                                                    case
20:Event20TArray=currTArray;Event20DArray=currDArray;
                                                    if (response1>0){v=w/20;SetCtrlVal (multPanel,
MULTPNL_20HITNUM, v);}break;
                                                    case
21:Event21TArray=currTArray;Event21DArray=currDArray;
                                                    if (response1>0){v=w/21;SetCtrlVal (multPanel,
MULTPNL_21HITNUM, v);}break;
                                                    case
22:Event22TArray=currTArray;Event22DArray=currDArray;
                                                    if (response1>0){v=w/22;SetCtrlVal (multPanel,
MULTPNL_22HITNUM, v);}break;
                                                    case
23:Event23TArray=currTArray;Event23DArray=currDArray;
                                                    if (response1>0){v=w/23;SetCtrlVal (multPanel,
MULTPNL_23HITNUM, v);}break;
                                                    case
24:Event24TArray=currTArray;Event24DArray=currDArray;
                                                    if (response1>0){v=w/24;SetCtrlVal (multPanel,
MULTPNL_24HITNUM, v);}break;
                                                    case
25:Event25TArray=currTArray;Event25DArray=currDArray;
                                                    if (response1>0){v=w/25;SetCtrlVal (multPanel,
MULTPNL_25HITNUM, v);}break;
                                                    case
26:Event26TArray=currTArray;Event26DArray=currDArray;
                                                    if (response1>0){v=w/26;SetCtrlVal (multPanel,
MULTPNL_26HITNUM, v);}break;
```

```
                                                    case
27:Event27TArray=currTArray;Event27DArray=currDArray;
                                                    if (response1>0){v=w/27;SetCtrlVal (multPanel,
MULTPNL_27HITNUM, v);}break;
                                                    case
28:Event28TArray=currTArray;Event28DArray=currDArray;
                                                    if (response1>0){v=w/28;SetCtrlVal (multPanel,
MULTPNL_28HITNUM, v);}break;
                                                    case
29:Event29TArray=currTArray;Event29DArray=currDArray;
                                                    if (response1>0){v=w/29;SetCtrlVal (multPanel,
MULTPNL_29HITNUM, v);}break;
                                                    case
30:Event30TArray=currTArray;Event30DArray=currDArray;
                                                    if (response1>0){v=w/30;SetCtrlVal (multPanel,
MULTPNL_30HITNUM, v);}break;
                                                    case
31:Event31TArray=currTArray;Event31DArray=currDArray;
                                                    if (response1>0){v=w/31;SetCtrlVal (multPanel,
MULTPNL_31HITNUM, v);}break;
                                                    case
32:Event32TArray=currTArray;Event32DArray=currDArray;
                                                    if (response1>0){v=w/32;SetCtrlVal (multPanel,
MULTPNL_32HITNUM, v);}break;
                                                    case
33:Event33TArray=currTArray;Event33DArray=currDArray;
                                                    if (response1>0){v=w/33;SetCtrlVal (multPanel,
MULTPNL_33HITNUM, v);}break;
                                                    case
34:Event34TArray=currTArray;Event34DArray=currDArray;
                                                    if (response1>0){v=w/34;SetCtrlVal (multPanel,
MULTPNL_34HITNUM, v);}break;
                                                    case
35:Event35TArray=currTArray;Event35DArray=currDArray;
                                                    if (response1>0){v=w/35;SetCtrlVal (multPanel,
MULTPNL_35HITNUM, v);}break;
                                                    case
36:Event36TArray=currTArray;Event36DArray=currDArray;
                                                    if (response1>0){v=w/36;SetCtrlVal (multPanel,
MULTPNL_36HITNUM, v);}break;
                                                    case
37:Event37TArray=currTArray;Event37DArray=currDArray;
                                                    if (response1>0){v=w/37;SetCtrlVal (multPanel,
MULTPNL_37HITNUM, v);}break;
                                                    case
38:Event38TArray=currTArray;Event38DArray=currDArray;
                                                    if (response1>0){v=w/38;SetCtrlVal (multPanel,
MULTPNL_38HITNUM, v);}break;
                                                    case
39:Event39TArray=currTArray;Event39DArray=currDArray;
                                                    if (response1>0){v=w/39;SetCtrlVal (multPanel,
MULTPNL_39HITNUM, v);}break;
```

```
                                                    case
40:Event40TArray=currTArray;Event40DArray=currDArray;
                                                    if (response1>0){v=w/40;SetCtrlVal (multPanel,
MULTPNL_40HITNUM, v);}break;
                                                    case
41:Event41TArray=currTArray;Event41DArray=currDArray;
                                                    if (response1>0){v=w/41;SetCtrlVal (multPanel,
MULTPNL_41HITNUM, v);}break;
                                                    case
42:Event42TArray=currTArray;Event42DArray=currDArray;
                                                    if (response1>0){v=w/42;SetCtrlVal (multPanel,
MULTPNL_42HITNUM, v);}break;
                                                    case
43:Event43TArray=currTArray;Event43DArray=currDArray;
                                                    if (response1>0){v=w/43;SetCtrlVal (multPanel,
MULTPNL_43HITNUM, v);}break;
                                                    case
44:Event44TArray=currTArray;Event44DArray=currDArray;
                                                    if (response1>0){v=w/44;SetCtrlVal (multPanel,
MULTPNL_44HITNUM, v);}break;
                                                    case
45:Event45TArray=currTArray;Event45DArray=currDArray;
                                                    if (response1>0){v=w/45;SetCtrlVal (multPanel,
MULTPNL_45HITNUM, v);}break;
                                                    case
46:Event46TArray=currTArray;Event46DArray=currDArray;
                                                    if (response1>0){v=w/46;SetCtrlVal (multPanel,
MULTPNL_46HITNUM, v);}break;
                                                    case
47:Event47TArray=currTArray;Event47DArray=currDArray;
                                                    if (response1>0){v=w/47;SetCtrlVal (multPanel,
MULTPNL_47HITNUM, v);}break;
                                                    case
48:Event48TArray=currTArray;Event48DArray=currDArray;
                                                    if (response1>0){v=w/48;SetCtrlVal (multPanel,
MULTPNL_48HITNUM, v);}break;
                                                    case
49:Event49TArray=currTArray;Event49DArray=currDArray;
                                                    if (response1>0){v=w/49;SetCtrlVal (multPanel,
MULTPNL_49HITNUM, v);}break;
                                                    case
50:Event50TArray=currTArray;Event50DArray=currDArray;
                                                    if (response1>0){v=w/50;SetCtrlVal (multPanel,
MULTPNL_50HITNUM, v);}break;

                                            }//switch

                                            }//void
void shiftData() {

        int x, y, z, q, w, v = 0;

                                    w=0;
```

```
                    for(x=0;x<=((nHit*n)-1);x++){

                            for(y=0;y<=(n-1);y++)    {

                                    tempT[y]=currTArray[x];
                                    tempD[y]=currDArray[x];
                                     x++;
                                                                    }
                            for(y=0;y<=(n-1);y++){
                                                                    if
((tempD[y] == 32768) || (tempD[y] == 0))    {tempD[y] = 0;}
                                                                    if
((tempD[y] == 33792) || (tempD[y] == 4096)) {tempD[y] = 1;}
                                                                    if
((tempD[y] == 34816) || (tempD[y] == 8192)) {tempD[y] = 2;}
                                                                    if
((tempD[y] == 35840) || (tempD[y] == 12288)){tempD[y] = 3;}
                                                                    if
((tempD[y] == 36864) || (tempD[y] == 16384)){tempD[y] = 4;}
                                                                    if
((tempD[y] == 37888) || (tempD[y] == 20480)){tempD[y] = 5;}
                                                                    if
((tempD[y] == 38912) || (tempD[y] == 24576)){tempD[y] = 6;}
                                                                    if
((tempD[y] == 39936) || (tempD[y] == 28672)){tempD[y] = 7;}

                                                                    }

                            for(z=0;z<=(n-1);z++){

                                                    if((tempT[z]<=ulm) &&
(tempT[z]>=llm)&&tempD[z]==en){
                                                    shift=tempT[z]-llm;

                    for(q=0;q<=(n-1);q++){


                                    det=tempD[q];

                            d=tempT[q]-shift;

                            if (d>0){



                    switch (det)

                            {
```

```
                case 0 : coinspec0[d]+=1;  break;
                case 1 : coinspec1[d]+=1;  break;
                case 2 : coinspec2[d]+=1;  break;
                case 3 : coinspec3[d]+=1;  break;
                case 4 : coinspec4[d]+=1;  break;
                case 5 : coinspec5[d]+=1;  break;
                case 6 : coinspec6[d]+=1;  break;
                case 7 : coinspec7[d]+=1;  break;
        }//switch
        }//if
        if(response1>0){

                currTArray[w]=d;
                currDArray[w]=det;
                w++;

        }
                }//for
z=n;
                }//if

                                                }//for
                x--;
                        }//for
        switch (n)
                {
```

```
                                                    case
1:Event1TArray=currTArray;Event1DArray=currDArray;
                                                    if (response1>0){v=w;SetCtrlVal (multPanel,
MULTPNL_1HITNUM, v);}break;
                                                    case
2:Event2TArray=currTArray;Event2DArray=currDArray;
                                                    if (response1>0){v=w/2;SetCtrlVal (multPanel,
MULTPNL_2HITNUM, v);}break;
                                                    case
3:Event3TArray=currTArray;Event3DArray=currDArray;
                                                    if (response1>0){v=w/3;SetCtrlVal (multPanel,
MULTPNL_3HITNUM, v);}break;
                                                    case
4:Event4TArray=currTArray;Event4DArray=currDArray;
                                                    if (response1>0){v=w/4;SetCtrlVal (multPanel,
MULTPNL_4HITNUM, v);}break;
                                                    case
5:Event5TArray=currTArray;Event5DArray=currDArray;
                                                    if (response1>0){v=w/5;SetCtrlVal (multPanel,
MULTPNL_5HITNUM, v);}break;
                                                    case
6:Event6TArray=currTArray;Event6DArray=currDArray;
                                                    if (response1>0){v=w/6;SetCtrlVal (multPanel,
MULTPNL_6HITNUM, v);}break;
                                                    case
7:Event7TArray=currTArray;Event7DArray=currDArray;
                                                    if (response1>0){v=w/7;SetCtrlVal (multPanel,
MULTPNL_7HITNUM, v);}break;
                                                    case
8:Event8TArray=currTArray;Event8DArray=currDArray;
                                                    if (response1>0){v=w/8;SetCtrlVal (multPanel,
MULTPNL_8HITNUM, v);}break;
                                                    case
9:Event9TArray=currTArray;Event9DArray=currDArray;
                                                    if (response1>0){v=w/9;SetCtrlVal (multPanel,
MULTPNL_9HITNUM, v);}break;
                                                    case
10:Event10TArray=currTArray;Event10DArray=currDArray;
                                                    if (response1>0){v=w/10;SetCtrlVal (multPanel,
MULTPNL_10HITNUM, v);}break;
                                                    case
11:Event11TArray=currTArray;Event11DArray=currDArray;
                                                    if (response1>0){v=w/11;SetCtrlVal (multPanel,
MULTPNL_11HITNUM, v);}break;
                                                    case
12:Event12TArray=currTArray;Event12DArray=currDArray;
                                                    if (response1>0){v=w/12;SetCtrlVal (multPanel,
MULTPNL_12HITNUM, v);}break;
                                                    case
13:Event13TArray=currTArray;Event13DArray=currDArray;
                                                    if (response1>0){v=w/13;SetCtrlVal (multPanel,
MULTPNL_13HITNUM, v);}break;
```

```
                                                    case
14:Event14TArray=currTArray;Event14DArray=currDArray;
                                                    if (response1>0){v=w/14;SetCtrlVal (multPanel,
MULTPNL_14HITNUM, v);}break;
                                                    case
15:Event15TArray=currTArray;Event15DArray=currDArray;
                                                    if (response1>0){v=w/15;SetCtrlVal (multPanel,
MULTPNL_15HITNUM, v);}break;
                                                    case
16:Event16TArray=currTArray;Event16DArray=currDArray;
                                                    if (response1>0){v=w/16;SetCtrlVal (multPanel,
MULTPNL_16HITNUM, v);}break;
                                                    case
17:Event17TArray=currTArray;Event17DArray=currDArray;
                                                    if (response1>0){v=w/17;SetCtrlVal (multPanel,
MULTPNL_17HITNUM, v);}break;
                                                    case
18:Event18TArray=currTArray;Event18DArray=currDArray;
                                                    if (response1>0){v=w/18;SetCtrlVal (multPanel,
MULTPNL_18HITNUM, v);}break;
                                                    case
19:Event19TArray=currTArray;Event19DArray=currDArray;
                                                    if (response1>0){v=w/19;SetCtrlVal (multPanel,
MULTPNL_19HITNUM, v);}break;
                                                    case
20:Event20TArray=currTArray;Event20DArray=currDArray;
                                                    if (response1>0){v=w/20;SetCtrlVal (multPanel,
MULTPNL_20HITNUM, v);}break;
                                                    case
21:Event21TArray=currTArray;Event21DArray=currDArray;
                                                    if (response1>0){v=w/21;SetCtrlVal (multPanel,
MULTPNL_21HITNUM, v);}break;
                                                    case
22:Event22TArray=currTArray;Event22DArray=currDArray;
                                                    if (response1>0){v=w/22;SetCtrlVal (multPanel,
MULTPNL_22HITNUM, v);}break;
                                                    case
23:Event23TArray=currTArray;Event23DArray=currDArray;
                                                    if (response1>0){v=w/23;SetCtrlVal (multPanel,
MULTPNL_23HITNUM, v);}break;
                                                    case
24:Event24TArray=currTArray;Event24DArray=currDArray;
                                                    if (response1>0){v=w/24;SetCtrlVal (multPanel,
MULTPNL_24HITNUM, v);}break;
                                                    case
25:Event25TArray=currTArray;Event25DArray=currDArray;
                                                    if (response1>0){v=w/25;SetCtrlVal (multPanel,
MULTPNL_25HITNUM, v);}break;
                                                    case
26:Event26TArray=currTArray;Event26DArray=currDArray;
                                                    if (response1>0){v=w/26;SetCtrlVal (multPanel,
MULTPNL_26HITNUM, v);}break;
```

```
                                                    case
27:Event27TArray=currTArray;Event27DArray=currDArray;
                                                    if (response1>0){v=w/27;SetCtrlVal (multPanel,
MULTPNL_27HITNUM, v);}break;
                                                    case
28:Event28TArray=currTArray;Event28DArray=currDArray;
                                                    if (response1>0){v=w/28;SetCtrlVal (multPanel,
MULTPNL_28HITNUM, v);}break;
                                                    case
29:Event29TArray=currTArray;Event29DArray=currDArray;
                                                    if (response1>0){v=w/29;SetCtrlVal (multPanel,
MULTPNL_29HITNUM, v);}break;
                                                    case
30:Event30TArray=currTArray;Event30DArray=currDArray;
                                                    if (response1>0){v=w/30;SetCtrlVal (multPanel,
MULTPNL_30HITNUM, v);}break;
                                                    case
31:Event31TArray=currTArray;Event31DArray=currDArray;
                                                    if (response1>0){v=w/31;SetCtrlVal (multPanel,
MULTPNL_31HITNUM, v);}break;
                                                    case
32:Event32TArray=currTArray;Event32DArray=currDArray;
                                                    if (response1>0){v=w/32;SetCtrlVal (multPanel,
MULTPNL_32HITNUM, v);}break;
                                                    case
33:Event33TArray=currTArray;Event33DArray=currDArray;
                                                    if (response1>0){v=w/33;SetCtrlVal (multPanel,
MULTPNL_33HITNUM, v);}break;
                                                    case
34:Event34TArray=currTArray;Event34DArray=currDArray;
                                                    if (response1>0){v=w/34;SetCtrlVal (multPanel,
MULTPNL_34HITNUM, v);}break;
                                                    case
35:Event35TArray=currTArray;Event35DArray=currDArray;
                                                    if (response1>0){v=w/35;SetCtrlVal (multPanel,
MULTPNL_35HITNUM, v);}break;
                                                    case
36:Event36TArray=currTArray;Event36DArray=currDArray;
                                                    if (response1>0){v=w/36;SetCtrlVal (multPanel,
MULTPNL_36HITNUM, v);}break;
                                                    case
37:Event37TArray=currTArray;Event37DArray=currDArray;
                                                    if (response1>0){v=w/37;SetCtrlVal (multPanel,
MULTPNL_37HITNUM, v);}break;
                                                    case
38:Event38TArray=currTArray;Event38DArray=currDArray;
                                                    if (response1>0){v=w/38;SetCtrlVal (multPanel,
MULTPNL_38HITNUM, v);}break;
                                                    case
39:Event39TArray=currTArray;Event39DArray=currDArray;
                                                    if (response1>0){v=w/39;SetCtrlVal (multPanel,
MULTPNL_39HITNUM, v);}break;
```

```
                                                    case
40:Event40TArray=currTArray;Event40DArray=currDArray;
                                                    if (response1>0){v=w/40;SetCtrlVal (multPanel,
MULTPNL_40HITNUM, v);}break;
                                                    case
41:Event41TArray=currTArray;Event41DArray=currDArray;
                                                    if (response1>0){v=w/41;SetCtrlVal (multPanel,
MULTPNL_41HITNUM, v);}break;
                                                    case
42:Event42TArray=currTArray;Event42DArray=currDArray;
                                                    if (response1>0){v=w/42;SetCtrlVal (multPanel,
MULTPNL_42HITNUM, v);}break;
                                                    case
43:Event43TArray=currTArray;Event43DArray=currDArray;
                                                    if (response1>0){v=w/43;SetCtrlVal (multPanel,
MULTPNL_43HITNUM, v);}break;
                                                    case
44:Event44TArray=currTArray;Event44DArray=currDArray;
                                                    if (response1>0){v=w/44;SetCtrlVal (multPanel,
MULTPNL_44HITNUM, v);}break;
                                                    case
45:Event45TArray=currTArray;Event45DArray=currDArray;
                                                    if (response1>0){v=w/45;SetCtrlVal (multPanel,
MULTPNL_45HITNUM, v);}break;
                                                    case
46:Event46TArray=currTArray;Event46DArray=currDArray;
                                                    if (response1>0){v=w/46;SetCtrlVal (multPanel,
MULTPNL_46HITNUM, v);}break;
                                                    case
47:Event47TArray=currTArray;Event47DArray=currDArray;
                                                    if (response1>0){v=w/47;SetCtrlVal (multPanel,
MULTPNL_47HITNUM, v);}break;
                                                    case
48:Event48TArray=currTArray;Event48DArray=currDArray;
                                                    if (response1>0){v=w/48;SetCtrlVal (multPanel,
MULTPNL_48HITNUM, v);}break;
                                                    case
49:Event49TArray=currTArray;Event49DArray=currDArray;
                                                    if (response1>0){v=w/49;SetCtrlVal (multPanel,
MULTPNL_49HITNUM, v);}break;
                                                    case
50:Event50TArray=currTArray;Event50DArray=currDArray;
                                                    if (response1>0){v=w/50;SetCtrlVal (multPanel,
MULTPNL_50HITNUM, v);}break;

                                                }//switch

                                                }//void

/************************************* Import RawData Program 8 CTN-M3
****************/
void CVICALLBACK MenuProcess8CB (int menuBar, int menuItem, void *callbackData,
                    int panel)
```

```
{
        int d = 0;
        unsigned int t1, t2, endtime, det, counter, f, x = 0;

         if(ConfirmPopup ("", "Load Experimental Parameters?")>0){

          FileSelectPopup (proj_dir, "*.exp", ".exp",
                                                  "Select Experimental Parameter File",
                                                  VAL_LOAD_BUTTON, 0, 1, 1, 1, file_name);
                ReadFileHandle = OpenFile (file_name, VAL_READ_ONLY, VAL_OPEN_AS_IS,
                                                                  VAL_BINARY);
                ScanFile (ReadFileHandle, "%s\n %s\n %s\n %s\n %s\n %s\n %s\n %s\n %s\n %s\n
%s\n %s\n %d\n",
                id, Date, Time, Date2, Time2, operator,target,pidat, sidat, startdat, stop1dat, stop2dat,
&chanmax);

                SetCtrlVal (expPnl, EXP_PNL_TM1TXTBOX, Time);
                SetCtrlVal (expPnl, EXP_PNL_DT1TXTBOX, Date);
                SetCtrlVal (expPnl, EXP_PNL_TM2TXTBOX, Time2);
                SetCtrlVal (expPnl, EXP_PNL_DT2TXTBOX, Date2);
                SetCtrlVal (expPnl, EXP_PNL_OPTEXTBOX, operator);
                SetCtrlVal (expPnl, EXP_PNL_IDTEXTBOX, id);
                SetCtrlVal (expPnl, EXP_PNL_TRTEXTBOX, target);
                SetCtrlVal (expPnl, EXP_PNL_PITEXTBOX, pidat);
                SetCtrlVal (expPnl, EXP_PNL_SIKTXTBOX, sidat);
                SetCtrlVal (expPnl, EXP_PNL_STDTXTBOX, startdat);
                SetCtrlVal (expPnl, EXP_PNL_ST1TXTBOX, stop1dat);
                SetCtrlVal (expPnl, EXP_PNL_ST2TXTBOX, stop2dat);
                SetCtrlVal (expPnl, EXP_PNL_MAXCHAN, chanmax);
                DisplayPanel(expPnl);


                                }//confirm

         if(FileSelectPopup ("", "*.CM3", "*.CM3", "Name of File to Read",
                                                  VAL_LOAD_BUTTON, 0, 0, 1, 1, file_name)>0){

         GetFileSize(file_name, &size);
         HidePanel(expPnl);

                histArray((size/2));
                FileToArray (file_name, array, VAL_UNSIGNED_SHORT_INTEGER,
                                                                  (size/2), 1,
VAL_GROUPS_TOGETHER,

VAL_GROUPS_AS_COLUMNS, VAL_BINARY);

                top:
                while (d<=((size/2)-1)){
```

```
                                        while( ((array[d]&64512) != 51200)  &&
(array[d]>0) ){
                                                det =
array[d]&64512;
                                                d++;
                                                switch (det)
                                                        {
                                                        case
32768 :
        ch0+=1;
        break;
                                                        case
33792 :
        ch1++;
        break;
                                                        case
34816 :
        ch2++;
        break;
                                                        case
35840 :
        ch3++;
        break;
                                                        case
36864 :
        ch4++;
        break;
                                                        case
37888 :
        ch5++;
        break;
                                                        case
38912 :
        ch6++;
        break;
                                                        case
39936 :
```

```
            ch7++;
            break;
                                                                    default:
            d++;
            counter = 0;
            goto top;
            break;
            }
                                                    if((array[d]==65535)||(array[d] ==
0)||(array[d]>32767)){
                                                            counter = 0;
                                                            d++;
                                                            goto top;
                                                                    }

                                                    counter++;
                                                    d++;

                                            }//while(Aarray[d]&64512 != 51200)
                                                    if((array[d]==65535)||(array[d] == 0)){
                                                            counter = 0;
                                                            d++;
                                                            goto top;
                                                                    }

                                                switch (counter)
                                                            {
                                                            case 0:totalEvents+=1;d++;d++;break;
                                                            case 1:stop1Events +=1;counter = 0;break;
                                                            case 2:stop2Events +=1;counter = 0;break;
                                                            case 3:stop3Events +=1;counter = 0;break;
                                                            case 4:stop4Events +=1;counter = 0;break;
                                                            case 5:stop5Events +=1;counter = 0;break;
                                                            case 6:stop6Events +=1;counter = 0;break;
                                                            case 7:stop7Events +=1;counter = 0;break;
                                                            case 8:stop8Events +=1;counter = 0;break;
                                                            case 9:stop9Events +=1;counter = 0;break;
                                                            case 10:stop10Events +=1;counter =
0;break;
                                                            case 11:stop11Events +=1;counter =
0;break;
                                                            case 12:stop12Events +=1;counter =
0;break;
```

```
                                                    case 13:stop13Events +=1;counter =
0;break;
                                                    case 14:stop14Events +=1;counter =
0;break;
                                                    case 15:stop15Events +=1;counter =
0;break;
                                                    case 16:stop16Events +=1;counter =
0;break;
                                                    case 17:stop17Events +=1;counter =
0;break;
                                                    case 18:stop18Events +=1;counter =
0;break;
                                                    case 19:stop19Events +=1;counter =
0;break;
                                                    case 20:stop20Events +=1;counter =
0;break;
                                                    case 21:stop21Events +=1;counter =
0;break;
                                                    case 22:stop22Events +=1;counter =
0;break;
                                                    case 23:stop23Events +=1;counter =
0;break;
                                                    case 24:stop24Events +=1;counter =
0;break;
                                                    case 25:stop25Events +=1;counter =
0;break;
                                                    case 26:stop26Events +=1;counter =
0;break;
                                                    case 27:stop27Events +=1;counter =
0;break;
                                                    case 28:stop28Events +=1;counter =
0;break;
                                                    case 29:stop29Events +=1;counter =
0;break;
                                                    case 30:stop30Events +=1;counter =
0;break;
                                                    case 31:stop31Events +=1;counter =
0;break;
                                                    case 32:stop32Events +=1;counter =
0;break;
                                                    case 33:stop33Events +=1;counter =
0;break;
                                                    case 34:stop34Events +=1;counter =
0;break;
                                                    case 35:stop35Events +=1;counter =
0;break;
                                                    case 36:stop36Events +=1;counter =
0;break;
                                                    case 37:stop37Events +=1;counter =
0;break;
                                                    case 38:stop38Events +=1;counter =
0;break;
```

```
                                                    case 39:stop39Events +=1;counter =
0;break;
                                                    case 40:stop40Events +=1;counter =
0;break;
                                                    case 41:stop41Events +=1;counter =
0;break;
                                                    case 42:stop42Events +=1;counter =
0;break;
                                                    case 43:stop43Events +=1;counter =
0;break;
                                                    case 44:stop44Events +=1;counter =
0;break;
                                                    case 45:stop45Events +=1;counter =
0;break;
                                                    case 46:stop46Events +=1;counter =
0;break;
                                                    case 47:stop47Events +=1;counter =
0;break;
                                                    case 48:stop48Events +=1;counter =
0;break;
                                                    case 49:stop49Events +=1;counter =
0;break;
                                                    case 50:stop50Events +=1;counter =
0;break;
                                                    default:counter=0;bigEvents+=1;break;
                                                    } // switch(counter)


                                                    }//while (array[d] != 0 )
                                        nullEvents = totalEvents-
(stop1Events+stop2Events+stop3Events+stop4Events+stop5Events+stop6Events+stop7Events+stop8Even
ts+stop9Events+stop10Events+

        stop11Events+stop12Events+stop13Events+stop14Events+stop15Events+stop16Events+stop17E
vents+stop18Events+stop19Events+stop20Events+stop21Events+

        stop22Events+stop23Events+stop24Events+stop25Events+stop26Events+stop27Events+stop28E
vents+stop29Events+stop30Events+stop31Events+stop32Events+stop33Events

        +stop34Events+stop35Events+stop36Events+stop37Events+stop38Events+stop39Events+stop40
Events+stop41Events+stop42Events+stop43Events

        +stop44Events+stop45Events+stop46Events+stop47Events+stop48Events+stop49Events+stop50
Events+bigEvents);


                                SetCtrlVal (multPanel, MULTPNL_TOTNUM, totalEvents);
                                SetCtrlVal (multPanel, MULTPNL_NULLNUM, nullEvents);
                                SetCtrlVal (multPanel, MULTPNL_1HITNUM, stop1Events);
                                SetCtrlVal (multPanel, MULTPNL_2HITNUM, stop2Events);
                                SetCtrlVal (multPanel, MULTPNL_3HITNUM, stop3Events);
                                SetCtrlVal (multPanel, MULTPNL_4HITNUM, stop4Events);
                                SetCtrlVal (multPanel, MULTPNL_5HITNUM, stop5Events);
```

```
SetCtrlVal (multPanel, MULTPNL_6HITNUM, stop6Events);
SetCtrlVal (multPanel, MULTPNL_7HITNUM, stop7Events);
SetCtrlVal (multPanel, MULTPNL_8HITNUM, stop8Events);
SetCtrlVal (multPanel, MULTPNL_9HITNUM, stop9Events);
SetCtrlVal (multPanel, MULTPNL_10HITNUM, stop10Events);
SetCtrlVal (multPanel, MULTPNL_11HITNUM, stop11Events);
SetCtrlVal (multPanel, MULTPNL_12HITNUM, stop12Events);
SetCtrlVal (multPanel, MULTPNL_13HITNUM, stop13Events);
SetCtrlVal (multPanel, MULTPNL_14HITNUM, stop14Events);
SetCtrlVal (multPanel, MULTPNL_15HITNUM, stop15Events);
SetCtrlVal (multPanel, MULTPNL_16HITNUM, stop16Events);
SetCtrlVal (multPanel, MULTPNL_17HITNUM, stop17Events);
SetCtrlVal (multPanel, MULTPNL_18HITNUM, stop18Events);
SetCtrlVal (multPanel, MULTPNL_19HITNUM, stop19Events);
SetCtrlVal (multPanel, MULTPNL_20HITNUM, stop20Events);
SetCtrlVal (multPanel, MULTPNL_21HITNUM, stop21Events);
SetCtrlVal (multPanel, MULTPNL_22HITNUM, stop22Events);
SetCtrlVal (multPanel, MULTPNL_23HITNUM, stop23Events);
SetCtrlVal (multPanel, MULTPNL_24HITNUM, stop24Events);
SetCtrlVal (multPanel, MULTPNL_25HITNUM, stop25Events);
SetCtrlVal (multPanel, MULTPNL_26HITNUM, stop26Events);
SetCtrlVal (multPanel, MULTPNL_27HITNUM, stop27Events);
SetCtrlVal (multPanel, MULTPNL_28HITNUM, stop28Events);
SetCtrlVal (multPanel, MULTPNL_29HITNUM, stop29Events);
SetCtrlVal (multPanel, MULTPNL_30HITNUM, stop30Events);
SetCtrlVal (multPanel, MULTPNL_31HITNUM, stop31Events);
SetCtrlVal (multPanel, MULTPNL_32HITNUM, stop32Events);
SetCtrlVal (multPanel, MULTPNL_33HITNUM, stop33Events);
SetCtrlVal (multPanel, MULTPNL_34HITNUM, stop34Events);
SetCtrlVal (multPanel, MULTPNL_35HITNUM, stop35Events);
SetCtrlVal (multPanel, MULTPNL_36HITNUM, stop36Events);
SetCtrlVal (multPanel, MULTPNL_37HITNUM, stop37Events);
SetCtrlVal (multPanel, MULTPNL_38HITNUM, stop38Events);
SetCtrlVal (multPanel, MULTPNL_39HITNUM, stop39Events);
SetCtrlVal (multPanel, MULTPNL_40HITNUM, stop40Events);
SetCtrlVal (multPanel, MULTPNL_41HITNUM, stop41Events);
SetCtrlVal (multPanel, MULTPNL_42HITNUM, stop42Events);
SetCtrlVal (multPanel, MULTPNL_43HITNUM, stop43Events);
SetCtrlVal (multPanel, MULTPNL_44HITNUM, stop44Events);
SetCtrlVal (multPanel, MULTPNL_45HITNUM, stop45Events);
SetCtrlVal (multPanel, MULTPNL_46HITNUM, stop46Events);
SetCtrlVal (multPanel, MULTPNL_47HITNUM, stop47Events);
SetCtrlVal (multPanel, MULTPNL_48HITNUM, stop48Events);
SetCtrlVal (multPanel, MULTPNL_49HITNUM, stop49Events);
SetCtrlVal (multPanel, MULTPNL_50HITNUM, stop50Events);

d=0;
endtime=0;
det=0;
counter=0;
f = 0;
t1=0;
t2=0;
```

```
if(stop1Events>0){
histAllocateEA1(stop1Events);SetCtrlAttribute (cnspecPnl, COINPNL_AR1_SW, ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES1, ATTR_DIMMED, 0);}
if(stop2Events>0){
histAllocateEA2(stop2Events*2);SetCtrlAttribute (cnspecPnl, COINPNL_AR2_SW, ATTR_DIMMED,
0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES2, ATTR_DIMMED, 0);}
if(stop3Events>0){
histAllocateEA3(stop3Events*3);SetCtrlAttribute (cnspecPnl, COINPNL_AR3_SW, ATTR_DIMMED,
0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES3, ATTR_DIMMED, 0);}
if(stop4Events>0){
histAllocateEA4(stop4Events*4);SetCtrlAttribute (cnspecPnl, COINPNL_AR4_SW, ATTR_DIMMED,
0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES4, ATTR_DIMMED, 0);}
if(stop5Events>0){
histAllocateEA5(stop5Events*5);SetCtrlAttribute (cnspecPnl, COINPNL_AR5_SW, ATTR_DIMMED,
0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES5, ATTR_DIMMED, 0);}
if(stop6Events>0){
histAllocateEA6(stop6Events*6);SetCtrlAttribute (cnspecPnl, COINPNL_AR6_SW, ATTR_DIMMED,
0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES6, ATTR_DIMMED, 0);}
if(stop7Events>0){
histAllocateEA7(stop7Events*7);SetCtrlAttribute (cnspecPnl, COINPNL_AR7_SW, ATTR_DIMMED,
0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES7, ATTR_DIMMED, 0);}
if(stop8Events>0){
histAllocateEA8(stop8Events*8);SetCtrlAttribute (cnspecPnl, COINPNL_AR8_SW, ATTR_DIMMED,
0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES8, ATTR_DIMMED, 0);}
if(stop9Events>0){
histAllocateEA9(stop9Events*9);SetCtrlAttribute (cnspecPnl, COINPNL_AR9_SW, ATTR_DIMMED,
0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES9, ATTR_DIMMED, 0);}
if(stop10Events>0){
histAllocateEA10(stop10Events*10);SetCtrlAttribute (cnspecPnl, COINPNL_AR10_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES10, ATTR_DIMMED, 0);}
if(stop11Events>0){
histAllocateEA11(stop11Events*11);SetCtrlAttribute (cnspecPnl, COINPNL_AR11_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES11, ATTR_DIMMED, 0);}
if(stop12Events>0){
histAllocateEA12(stop12Events*12);SetCtrlAttribute (cnspecPnl, COINPNL_AR12_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES12, ATTR_DIMMED, 0);}
if(stop13Events>0){
histAllocateEA13(stop13Events*13);SetCtrlAttribute (cnspecPnl, COINPNL_AR13_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES13, ATTR_DIMMED, 0);}
if(stop14Events>0){
```

```
histAllocateEA14(stop14Events*14);SetCtrlAttribute (cnspecPnl, COINPNL_AR14_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES14, ATTR_DIMMED, 0);}
if(stop15Events>0){
histAllocateEA15(stop15Events*15);SetCtrlAttribute (cnspecPnl, COINPNL_AR15_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES15, ATTR_DIMMED, 0);}
if(stop16Events>0){
histAllocateEA16(stop16Events*16);SetCtrlAttribute (cnspecPnl, COINPNL_AR16_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES16, ATTR_DIMMED, 0);}
if(stop17Events>0){
histAllocateEA17(stop17Events*17);SetCtrlAttribute (cnspecPnl, COINPNL_AR17_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES17, ATTR_DIMMED, 0);}
if(stop18Events>0){
histAllocateEA18(stop18Events*18);SetCtrlAttribute (cnspecPnl, COINPNL_AR18_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES18, ATTR_DIMMED, 0);}
if(stop19Events>0){
histAllocateEA19(stop19Events*19);SetCtrlAttribute (cnspecPnl, COINPNL_AR19_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES19, ATTR_DIMMED, 0);}
if(stop20Events>0){
histAllocateEA20(stop20Events*20);SetCtrlAttribute (cnspecPnl, COINPNL_AR20_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES20, ATTR_DIMMED, 0);}
if(stop21Events>0){
histAllocateEA21(stop21Events*21);SetCtrlAttribute (cnspecPnl, COINPNL_AR21_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES21, ATTR_DIMMED, 0);}
if(stop22Events>0){
histAllocateEA22(stop22Events*22);SetCtrlAttribute (cnspecPnl, COINPNL_AR22_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES22, ATTR_DIMMED, 0);}
if(stop23Events>0){
histAllocateEA23(stop23Events*23);SetCtrlAttribute (cnspecPnl, COINPNL_AR23_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES23, ATTR_DIMMED, 0);}
if(stop24Events>0){
histAllocateEA24(stop24Events*24);SetCtrlAttribute (cnspecPnl, COINPNL_AR24_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES24, ATTR_DIMMED, 0);}
if(stop25Events>0){
histAllocateEA25(stop25Events*25);SetCtrlAttribute (cnspecPnl, COINPNL_AR25_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES25, ATTR_DIMMED, 0);}
if(stop26Events>0){
histAllocateEA26(stop26Events*26);SetCtrlAttribute (cnspecPnl, COINPNL_AR26_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES26, ATTR_DIMMED, 0);}
if(stop27Events>0){
```

```
histAllocateEA27(stop27Events*27);SetCtrlAttribute (cnspecPnl, COINPNL_AR27_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES27, ATTR_DIMMED, 0);}
if(stop28Events>0){
histAllocateEA28(stop28Events*28);SetCtrlAttribute (cnspecPnl, COINPNL_AR28_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES28, ATTR_DIMMED, 0);}
if(stop29Events>0){
histAllocateEA29(stop29Events*29);SetCtrlAttribute (cnspecPnl, COINPNL_AR29_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES29, ATTR_DIMMED, 0);}
if(stop30Events>0){
histAllocateEA30(stop30Events*30);SetCtrlAttribute (cnspecPnl, COINPNL_AR30_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES30, ATTR_DIMMED, 0);}
if(stop31Events>0){
histAllocateEA31(stop31Events*31);SetCtrlAttribute (cnspecPnl, COINPNL_AR31_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES31, ATTR_DIMMED, 0);}
if(stop32Events>0){
histAllocateEA32(stop32Events*32);SetCtrlAttribute (cnspecPnl, COINPNL_AR32_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES32, ATTR_DIMMED, 0);}
if(stop33Events>0){
histAllocateEA33(stop33Events*33);SetCtrlAttribute (cnspecPnl, COINPNL_AR33_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES33, ATTR_DIMMED, 0);}
if(stop34Events>0){
histAllocateEA34(stop34Events*34);SetCtrlAttribute (cnspecPnl, COINPNL_AR34_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES34, ATTR_DIMMED, 0);}
if(stop35Events>0){
histAllocateEA35(stop35Events*35);SetCtrlAttribute (cnspecPnl, COINPNL_AR35_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES35, ATTR_DIMMED, 0);}
if(stop36Events>0){
histAllocateEA36(stop36Events*36);SetCtrlAttribute (cnspecPnl, COINPNL_AR36_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES36, ATTR_DIMMED, 0);}
if(stop37Events>0){
histAllocateEA37(stop37Events*37);SetCtrlAttribute (cnspecPnl, COINPNL_AR37_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES37, ATTR_DIMMED, 0);}
if(stop38Events>0){
histAllocateEA38(stop38Events*38);SetCtrlAttribute (cnspecPnl, COINPNL_AR38_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES38, ATTR_DIMMED, 0);}
if(stop39Events>0){
histAllocateEA39(stop39Events*39);SetCtrlAttribute (cnspecPnl, COINPNL_AR39_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES39, ATTR_DIMMED, 0);}
if(stop40Events>0){
```

```
histAllocateEA40(stop40Events*40);SetCtrlAttribute (cnspecPnl, COINPNL_AR40_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES40, ATTR_DIMMED, 0);}
if(stop41Events>0){
histAllocateEA41(stop41Events*41);SetCtrlAttribute (cnspecPnl, COINPNL_AR41_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES41, ATTR_DIMMED, 0);}
if(stop42Events>0){
histAllocateEA42(stop42Events*42);SetCtrlAttribute (cnspecPnl, COINPNL_AR42_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES42, ATTR_DIMMED, 0);}
if(stop43Events>0){
histAllocateEA43(stop43Events*43);SetCtrlAttribute (cnspecPnl, COINPNL_AR43_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES43, ATTR_DIMMED, 0);}
if(stop44Events>0){
histAllocateEA44(stop44Events*44);SetCtrlAttribute (cnspecPnl, COINPNL_AR44_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES44, ATTR_DIMMED, 0);}
if(stop45Events>0){
histAllocateEA45(stop45Events*45);SetCtrlAttribute (cnspecPnl, COINPNL_AR45_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES45, ATTR_DIMMED, 0);}
if(stop46Events>0){
histAllocateEA46(stop46Events*46);SetCtrlAttribute (cnspecPnl, COINPNL_AR46_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES46, ATTR_DIMMED, 0);}
if(stop47Events>0){
histAllocateEA47(stop47Events*47);SetCtrlAttribute (cnspecPnl, COINPNL_AR47_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES47, ATTR_DIMMED, 0);}
if(stop48Events>0){
histAllocateEA48(stop48Events*48);SetCtrlAttribute (cnspecPnl, COINPNL_AR48_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES48, ATTR_DIMMED, 0);}
if(stop49Events>0){
histAllocateEA49(stop49Events*49);SetCtrlAttribute (cnspecPnl, COINPNL_AR49_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES49, ATTR_DIMMED, 0);}
if(stop50Events>0){
histAllocateEA50(stop50Events*50);SetCtrlAttribute (cnspecPnl, COINPNL_AR50_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES50, ATTR_DIMMED, 0);}

if(ch0>0){
SetCtrlAttribute (cnspecPnl, COINPNL_CH0_SW, ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_CH0_SWEN, ATTR_DIMMED, 0);}
if(ch1>0){
SetCtrlAttribute (cnspecPnl, COINPNL_CH1_SW, ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_CH1_SWEN, ATTR_DIMMED, 0);}
if(ch2>0){
SetCtrlAttribute (cnspecPnl, COINPNL_CH2_SW, ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_CH2_SWEN, ATTR_DIMMED, 0);}
```

```
if(ch3>0){
SetCtrlAttribute (cnspecPnl, COINPNL_CH3_SW, ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_CH3_SWEN, ATTR_DIMMED, 0);}
if(ch4>0){
SetCtrlAttribute (cnspecPnl, COINPNL_CH4_SW, ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_CH4_SWEN, ATTR_DIMMED, 0);}
if(ch5>0){
SetCtrlAttribute (cnspecPnl, COINPNL_CH5_SW, ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_CH5_SWEN, ATTR_DIMMED, 0);}
if(ch6>0){
SetCtrlAttribute (cnspecPnl, COINPNL_CH6_SW, ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_CH6_SWEN, ATTR_DIMMED, 0);}
if(ch7>0){
SetCtrlAttribute (cnspecPnl, COINPNL_CH7_SW, ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_CH7_SWEN, ATTR_DIMMED, 0);}


                d=0;
                top1:
                while (d<=((size/2)-1)){

                                                while( ((array[d]&64512) != 51200)  &&
(array[d]>0) ){

                                                                                det =
array[d]&64512;
                                                                                t1 =
array[d]&255;
                                                                                d++;

                                                                                switch (det)
                                                                                                {
                                                                                                case
32768 :

        ch0+=1;

        break;
                                                                                case
33792 :

        ch1++;

        break;
                                                                                case
34816 :

        ch2++;

        break;
                                                                                case
35840 :
```

```
            ch3++;
            break;
                                                                    case
36864 :
            ch4++;
            break;
                                                                    case
37888 :
            ch5++;
            break;
                                                                    case
38912 :
            ch6++;
            break;
                                                                    case
39936 :
            ch7++;
            break;
                                                                    default:

            d++;
            counter = 0;
            goto top1;
            break;
            }
            if((array[d]==65535)||(array[d] == 0)){
                        counter = 0;
                        d++;
                        goto top1;
                                                }
```

```
                                                                    endtime =
array[d]+(t1*32768);
                                                                    counter++;
                                                                    d++;

                                                                    f=counter;

                                                                    tempT[f-1] =
endtime;
                                                                    tempD[f-1] =
det;


                        }//while(Aarray[d]&64512 != 51200)


        if((array[d]==65535)||(array[d] == 0)){

                counter = 0;

                d++;

                goto top1;

                                }

                                                switch (counter)
                                                        {

                                                        case 0:d++;d++;break;
                                                        case 1:for(x=0;x<=(counter-1);x++){

        Event1TArray[ea1]=tempT[x];Event1DArray[ea1]=tempD[x];ea1++;}
                                                        counter = 0;x=0;break;
                                                        case 2:for(x=0;x<=(counter-1);x++){

        Event2TArray[ea2]=tempT[x];Event2DArray[ea2]=tempD[x];ea2++;}
                                                        counter = 0;x=0;break;
                                                        case 3:for(x=0;x<=(counter-1);x++){

        Event3TArray[ea3]=tempT[x];Event3DArray[ea3]=tempD[x];ea3++;}
                                                        counter = 0;x=0;break;
                                                        case 4:for(x=0;x<=(counter-1);x++){

        Event4TArray[ea4]=tempT[x];Event4DArray[ea4]=tempD[x];ea4++;}
                                                        counter = 0;x=0;break;
                                                        case 5:for(x=0;x<=(counter-1);x++){

        Event5TArray[ea5]=tempT[x];Event5DArray[ea5]=tempD[x];ea5++;}
                                                        counter = 0;x=0;break;
                                                        case 6:for(x=0;x<=(counter-1);x++){
```

```
Event6TArray[ea6]=tempT[x];Event6DArray[ea6]=tempD[x];ea6++;}
                                                    counter = 0;x=0;break;
                                                    case 7:for(x=0;x<=(counter-1);x++){

Event7TArray[ea7]=tempT[x];Event7DArray[ea7]=tempD[x];ea7++;}
                                                    counter = 0;x=0;break;
                                                    case 8:for(x=0;x<=(counter-1);x++){

Event8TArray[ea8]=tempT[x];Event8DArray[ea8]=tempD[x];ea8++;}
                                                    counter = 0;x=0;break;
                                                    case 9:for(x=0;x<=(counter-1);x++){

Event9TArray[ea9]=tempT[x];Event9DArray[ea9]=tempD[x];ea9++;}
                                                    counter = 0;x=0;break;
                                                    case 10:for(x=0;x<=(counter-1);x++){

Event10TArray[ea10]=tempT[x];Event10DArray[ea10]=tempD[x];ea10++;}
                                                    counter = 0;x=0;break;
                                                    case 11:for(x=0;x<=(counter-1);x++){

Event11TArray[ea11]=tempT[x];Event11DArray[ea11]=tempD[x];ea11++;}
                                                    counter = 0;x=0;break;
                                                    case 12:for(x=0;x<=(counter-1);x++){

Event12TArray[ea12]=tempT[x];Event12DArray[ea12]=tempD[x];ea12++;}
                                                    counter = 0;x=0;break;
                                                    case 13:for(x=0;x<=(counter-1);x++){

Event13TArray[ea13]=tempT[x];Event13DArray[ea13]=tempD[x];ea13++;}
                                                    counter = 0;x=0;break;
                                                    case 14:for(x=0;x<=(counter-1);x++){

Event14TArray[ea14]=tempT[x];Event14DArray[ea14]=tempD[x];ea14++;}
                                                    counter = 0;x=0;break;
                                                    case 15:for(x=0;x<=(counter-1);x++){

Event15TArray[ea15]=tempT[x];Event15DArray[ea15]=tempD[x];ea15++;}
                                                    counter = 0;x=0;break;
                                                    case 16:for(x=0;x<=(counter-1);x++){

Event16TArray[ea16]=tempT[x];Event16DArray[ea16]=tempD[x];ea16++;}
                                                    counter = 0;x=0;break;
                                                    case 17:for(x=0;x<=(counter-1);x++){

Event17TArray[ea17]=tempT[x];Event17DArray[ea17]=tempD[x];ea17++;}
                                                    counter = 0;x=0;break;
                                                    case 18:for(x=0;x<=(counter-1);x++){

Event18TArray[ea18]=tempT[x];Event18DArray[ea18]=tempD[x];ea18++;}
                                                    counter = 0;x=0;break;
                                                    case 19:for(x=0;x<=(counter-1);x++){
```

```
Event19TArray[ea19]=tempT[x];Event19DArray[ea19]=tempD[x];ea19++;}
                                                    counter = 0;x=0;break;
                                                    case 20:for(x=0;x<=(counter-1);x++){

Event20TArray[ea20]=tempT[x];Event20DArray[ea20]=tempD[x];ea20++;}
                                                    counter = 0;x=0;break;
                                                    case 21:for(x=0;x<=(counter-1);x++){

Event21TArray[ea21]=tempT[x];Event21DArray[ea21]=tempD[x];ea21++;}
                                                    counter = 0;x=0;break;
                                                    case 22:for(x=0;x<=(counter-1);x++){

Event22TArray[ea22]=tempT[x];Event22DArray[ea22]=tempD[x];ea22++;}
                                                    counter = 0;x=0;break;
                                                    case 23:for(x=0;x<=(counter-1);x++){

Event23TArray[ea23]=tempT[x];Event23DArray[ea23]=tempD[x];ea23++;}
                                                    counter = 0;x=0;break;
                                                    case 24:for(x=0;x<=(counter-1);x++){

Event24TArray[ea24]=tempT[x];Event24DArray[ea24]=tempD[x];ea24++;}
                                                    counter = 0;x=0;break;
                                                    case 25:for(x=0;x<=(counter-1);x++){

Event25TArray[ea25]=tempT[x];Event25DArray[ea25]=tempD[x];ea25++;}
                                                    counter = 0;x=0;break;
                                                    case 26:for(x=0;x<=(counter-1);x++){

Event26TArray[ea26]=tempT[x];Event26DArray[ea26]=tempD[x];ea26++;}
                                                    counter = 0;x=0;break;
                                                    case 27:for(x=0;x<=(counter-1);x++){

Event27TArray[ea27]=tempT[x];Event27DArray[ea27]=tempD[x];ea27++;}
                                                    counter = 0;x=0;break;
                                                    case 28:for(x=0;x<=(counter-1);x++){

Event28TArray[ea28]=tempT[x];Event28DArray[ea28]=tempD[x];ea28++;}
                                                    counter = 0;x=0;break;
                                                    case 29:for(x=0;x<=(counter-1);x++){

Event29TArray[ea29]=tempT[x];Event29DArray[ea29]=tempD[x];ea29++;}
                                                    counter = 0;x=0;break;
                                                    case 30:for(x=0;x<=(counter-1);x++){

Event30TArray[ea30]=tempT[x];Event30DArray[ea30]=tempD[x];ea30++;}
                                                    counter = 0;x=0;break;
                                                    case 31:for(x=0;x<=(counter-1);x++){

Event31TArray[ea31]=tempT[x];Event31DArray[ea31]=tempD[x];ea31++;}
                                                    counter = 0;x=0;break;
                                                    case 32:for(x=0;x<=(counter-1);x++){
```

```
Event32TArray[ea32]=tempT[x];Event32DArray[ea32]=tempD[x];ea32++;}
                                                    counter = 0;x=0;break;
                                                    case 33:for(x=0;x<=(counter-1);x++){

Event33TArray[ea33]=tempT[x];Event33DArray[ea33]=tempD[x];ea33++;}
                                                    counter = 0;x=0;break;
                                                    case 34:for(x=0;x<=(counter-1);x++){

Event34TArray[ea34]=tempT[x];Event34DArray[ea34]=tempD[x];ea34++;}
                                                    counter = 0;x=0;break;
                                                    case 35:for(x=0;x<=(counter-1);x++){

Event35TArray[ea35]=tempT[x];Event35DArray[ea35]=tempD[x];ea35++;}
                                                    counter = 0;x=0;break;
                                                    case 36:for(x=0;x<=(counter-1);x++){

Event36TArray[ea36]=tempT[x];Event36DArray[ea36]=tempD[x];ea36++;}
                                                    counter = 0;x=0;break;
                                                    case 37:for(x=0;x<=(counter-1);x++){

Event37TArray[ea37]=tempT[x];Event37DArray[ea37]=tempD[x];ea37++;}
                                                    counter = 0;x=0;break;
                                                    case 38:for(x=0;x<=(counter-1);x++){

Event38TArray[ea38]=tempT[x];Event38DArray[ea38]=tempD[x];ea38++;}
                                                    counter = 0;x=0;break;
                                                    case 39:for(x=0;x<=(counter-1);x++){

Event39TArray[ea39]=tempT[x];Event39DArray[ea39]=tempD[x];ea39++;}
                                                    counter = 0;x=0;break;
                                                    case 40:for(x=0;x<=(counter-1);x++){

Event40TArray[ea40]=tempT[x];Event40DArray[ea40]=tempD[x];ea40++;}
                                                    counter = 0;x=0;break;
                                                    case 41:for(x=0;x<=(counter-1);x++){

Event41TArray[ea41]=tempT[x];Event41DArray[ea41]=tempD[x];ea41++;}
                                                    counter = 0;x=0;break;
                                                    case 42:for(x=0;x<=(counter-1);x++){

Event42TArray[ea42]=tempT[x];Event42DArray[ea42]=tempD[x];ea42++;}
                                                    counter = 0;x=0;break;
                                                    case 43:for(x=0;x<=(counter-1);x++){

Event43TArray[ea43]=tempT[x];Event43DArray[ea43]=tempD[x];ea43++;}
                                                    counter = 0;x=0;break;
                                                    case 44:for(x=0;x<=(counter-1);x++){

Event44TArray[ea44]=tempT[x];Event44DArray[ea44]=tempD[x];ea44++;}
                                                    counter = 0;x=0;break;
                                                    case 45:for(x=0;x<=(counter-1);x++){
```

```
        Event45TArray[ea45]=tempT[x];Event45DArray[ea45]=tempD[x];ea45++;}
                                                        counter = 0;x=0;break;
                                                        case 46:for(x=0;x<=(counter-1);x++){

        Event46TArray[ea46]=tempT[x];Event46DArray[ea46]=tempD[x];ea46++;}
                                                        counter = 0;x=0;break;
                                                        case 47:for(x=0;x<=(counter-1);x++){

        Event47TArray[ea47]=tempT[x];Event47DArray[ea47]=tempD[x];ea47++;}
                                                        counter = 0;x=0;break;
                                                        case 48:for(x=0;x<=(counter-1);x++){

        Event48TArray[ea48]=tempT[x];Event48DArray[ea48]=tempD[x];ea48++;}
                                                        counter = 0;x=0;break;
                                                        case 49:for(x=0;x<=(counter-1);x++){

        Event49TArray[ea49]=tempT[x];Event49DArray[ea49]=tempD[x];ea49++;}
                                                        counter = 0;x=0;break;
                                                        case 50:for(x=0;x<=(counter-1);x++){

        Event50TArray[ea50]=tempT[x];Event50DArray[ea50]=tempD[x];ea50++;}
                                                        counter = 0;x=0;break;

                                                        default:counter=0;break;
                                                        } // switch(counter)


          }//while (array[d] != 0 )
                                SetCtrlVal (mnPanel, MN_PANEL_NUMEVENT, totalEvents);

}
}
/***********************************Import RawData Program A CTN-
M4*********************/
void CVICALLBACK MenuProcessACB (int menuBar, int menuItem, void *callbackData,
                int panel)
{
        int d = 0;
        unsigned int t1, t2, endtime, det, counter, f, x = 0;

         if(ConfirmPopup ("", "Load Experimental Parameters?")>0){

          FileSelectPopup (proj_dir, "*.exp", ".exp",
                                                        "Select Experimental Parameter File",
                                                        VAL_LOAD_BUTTON, 0, 1, 1, 1, file_name);
                ReadFileHandle = OpenFile (file_name, VAL_READ_ONLY, VAL_OPEN_AS_IS,
                                                                VAL_BINARY);
                ScanFile (ReadFileHandle, "%s\n %s\n %s\n %s\n %s\n %s\n %s\n %s\n %s\n %s\n
%s\n %s\n %d\n",
                id, Date, Time, Date2, Time2, operator,target,pidat, sidat, startdat, stop1dat, stop2dat,
&chanmax);
```

```
                SetCtrlVal (expPnl, EXP_PNL_TM1TXTBOX, Time);
                SetCtrlVal (expPnl, EXP_PNL_DT1TXTBOX, Date);
                SetCtrlVal (expPnl, EXP_PNL_TM2TXTBOX, Time2);
                SetCtrlVal (expPnl, EXP_PNL_DT2TXTBOX, Date2);
                SetCtrlVal (expPnl, EXP_PNL_OPTEXTBOX, operator);
                SetCtrlVal (expPnl, EXP_PNL_IDTEXTBOX, id);
                SetCtrlVal (expPnl, EXP_PNL_TRTEXTBOX, target);
                SetCtrlVal (expPnl, EXP_PNL_PITEXTBOX, pidat);
                SetCtrlVal (expPnl, EXP_PNL_SIKTXTBOX, sidat);
                SetCtrlVal (expPnl, EXP_PNL_STDTXTBOX, startdat);
                SetCtrlVal (expPnl, EXP_PNL_ST1TXTBOX, stop1dat);
                SetCtrlVal (expPnl, EXP_PNL_ST2TXTBOX, stop2dat);
                SetCtrlVal (expPnl, EXP_PNL_MAXCHAN, chanmax);
                DisplayPanel(expPnl);


                                }//confirm

        if(FileSelectPopup ("", "*.CM4", "*.CM4", "Name of File to Read",
                                                VAL_LOAD_BUTTON, 0, 1, 1, 1, file_name)>0){

        GetFileSize(file_name, &size);
        HidePanel(expPnl);

                histArray((size/2));
                FileToArray (file_name, array, VAL_UNSIGNED_SHORT_INTEGER,
                                                                (size/2), 1,
VAL_GROUPS_TOGETHER,

VAL_GROUPS_AS_COLUMNS, VAL_BINARY);

                d=((size/2)-1);

                 top:
                while ( d >= 0){

                                        while( (d >=0)&&((array[d]&61440) != 32768)){

                                                                det =
(array[d]&61440);
                                                                --d;
                                                                switch (det)
                                                                        {
                                                                        case 0 :

        ch0+=1;

        break;
                                                                        case
4096 :

        ch1++;
```

```
            break;
                                                                    case
8192 :
            ch2++;
            break;
                                                                    case
12288 :
            ch3++;
            break;
                                                                    case
16384 :
            ch4++;
            break;
                                                                    case
20480 :
            ch5++;
            break;
                                                                    case
24576 :
            ch6++;
            break;
                                                                    case
28672 :
            ch7++;
            break;
                                                                    default:
            --d;
            counter = 0;
            goto top;
            break;
            }
                                                counter++;
                                                --d;
```

```
}//while(Aarray[d]&64512 != 51200)


                switch (counter)
                        {
                        case 0:totalEvents+=1;--d;--d;break;
                        case 1:stop1Events +=1;counter = 0;break;
                        case 2:stop2Events +=1;counter = 0;break;
                        case 3:stop3Events +=1;counter = 0;break;
                        case 4:stop4Events +=1;counter = 0;break;
                        case 5:stop5Events +=1;counter = 0;break;
                        case 6:stop6Events +=1;counter = 0;break;
                        case 7:stop7Events +=1;counter = 0;break;
                        case 8:stop8Events +=1;counter = 0;break;
                        case 9:stop9Events +=1;counter = 0;break;
                        case 10:stop10Events +=1;counter =
0;break;
                        case 11:stop11Events +=1;counter =
0;break;
                        case 12:stop12Events +=1;counter =
0;break;
                        case 13:stop13Events +=1;counter =
0;break;
                        case 14:stop14Events +=1;counter =
0;break;
                        case 15:stop15Events +=1;counter =
0;break;
                        case 16:stop16Events +=1;counter =
0;break;
                        case 17:stop17Events +=1;counter =
0;break;
                        case 18:stop18Events +=1;counter =
0;break;
                        case 19:stop19Events +=1;counter =
0;break;
                        case 20:stop20Events +=1;counter =
0;break;
                        case 21:stop21Events +=1;counter =
0;break;
                        case 22:stop22Events +=1;counter =
0;break;
                        case 23:stop23Events +=1;counter =
0;break;
                        case 24:stop24Events +=1;counter =
0;break;
                        case 25:stop25Events +=1;counter =
0;break;
                        case 26:stop26Events +=1;counter =
0;break;
                        case 27:stop27Events +=1;counter =
0;break;
```

```
case 28:stop28Events +=1;counter = 0;break;
case 29:stop29Events +=1;counter = 0;break;
case 30:stop30Events +=1;counter = 0;break;
case 31:stop31Events +=1;counter = 0;break;
case 32:stop32Events +=1;counter = 0;break;
case 33:stop33Events +=1;counter = 0;break;
case 34:stop34Events +=1;counter = 0;break;
case 35:stop35Events +=1;counter = 0;break;
case 36:stop36Events +=1;counter = 0;break;
case 37:stop37Events +=1;counter = 0;break;
case 38:stop38Events +=1;counter = 0;break;
case 39:stop39Events +=1;counter = 0;break;
case 40:stop40Events +=1;counter = 0;break;
case 41:stop41Events +=1;counter = 0;break;
case 42:stop42Events +=1;counter = 0;break;
case 43:stop43Events +=1;counter = 0;break;
case 44:stop44Events +=1;counter = 0;break;
case 45:stop45Events +=1;counter = 0;break;
case 46:stop46Events +=1;counter = 0;break;
case 47:stop47Events +=1;counter = 0;break;
case 48:stop48Events +=1;counter = 0;break;
case 49:stop49Events +=1;counter = 0;break;
case 50:stop50Events +=1;counter = 0;break;
default:counter=0;bigEvents+=1;break;
} // switch(counter)

}//while (array[d] != 0 )
```

```
nullEvents = totalEvents-
(stop1Events+stop2Events+stop3Events+stop4Events+stop5Events+stop6Events+stop7Events+stop8Even
ts+stop9Events+stop10Events+

    stop11Events+stop12Events+stop13Events+stop14Events+stop15Events+stop16Events+stop17E
vents+stop18Events+stop19Events+stop20Events+stop21Events+

    stop22Events+stop23Events+stop24Events+stop25Events+stop26Events+stop27Events+stop28E
vents+stop29Events+stop30Events+stop31Events+stop32Events+stop33Events

    +stop34Events+stop35Events+stop36Events+stop37Events+stop38Events+stop39Events+stop40
Events+stop41Events+stop42Events+stop43Events

    +stop44Events+stop45Events+stop46Events+stop47Events+stop48Events+stop49Events+stop50
Events+bigEvents);


        SetCtrlVal (multPanel, MULTPNL_TOTNUM, totalEvents);
        SetCtrlVal (multPanel, MULTPNL_NULLNUM, nullEvents);
        SetCtrlVal (multPanel, MULTPNL_1HITNUM, stop1Events);
        SetCtrlVal (multPanel, MULTPNL_2HITNUM, stop2Events);
        SetCtrlVal (multPanel, MULTPNL_3HITNUM, stop3Events);
        SetCtrlVal (multPanel, MULTPNL_4HITNUM, stop4Events);
        SetCtrlVal (multPanel, MULTPNL_5HITNUM, stop5Events);
        SetCtrlVal (multPanel, MULTPNL_6HITNUM, stop6Events);
        SetCtrlVal (multPanel, MULTPNL_7HITNUM, stop7Events);
        SetCtrlVal (multPanel, MULTPNL_8HITNUM, stop8Events);
        SetCtrlVal (multPanel, MULTPNL_9HITNUM, stop9Events);
        SetCtrlVal (multPanel, MULTPNL_10HITNUM, stop10Events);
        SetCtrlVal (multPanel, MULTPNL_11HITNUM, stop11Events);
        SetCtrlVal (multPanel, MULTPNL_12HITNUM, stop12Events);
        SetCtrlVal (multPanel, MULTPNL_13HITNUM, stop13Events);
        SetCtrlVal (multPanel, MULTPNL_14HITNUM, stop14Events);
        SetCtrlVal (multPanel, MULTPNL_15HITNUM, stop15Events);
        SetCtrlVal (multPanel, MULTPNL_16HITNUM, stop16Events);
        SetCtrlVal (multPanel, MULTPNL_17HITNUM, stop17Events);
        SetCtrlVal (multPanel, MULTPNL_18HITNUM, stop18Events);
        SetCtrlVal (multPanel, MULTPNL_19HITNUM, stop19Events);
        SetCtrlVal (multPanel, MULTPNL_20HITNUM, stop20Events);
        SetCtrlVal (multPanel, MULTPNL_21HITNUM, stop21Events);
        SetCtrlVal (multPanel, MULTPNL_22HITNUM, stop22Events);
        SetCtrlVal (multPanel, MULTPNL_23HITNUM, stop23Events);
        SetCtrlVal (multPanel, MULTPNL_24HITNUM, stop24Events);
        SetCtrlVal (multPanel, MULTPNL_25HITNUM, stop25Events);
        SetCtrlVal (multPanel, MULTPNL_26HITNUM, stop26Events);
        SetCtrlVal (multPanel, MULTPNL_27HITNUM, stop27Events);
        SetCtrlVal (multPanel, MULTPNL_28HITNUM, stop28Events);
        SetCtrlVal (multPanel, MULTPNL_29HITNUM, stop29Events);
        SetCtrlVal (multPanel, MULTPNL_30HITNUM, stop30Events);
        SetCtrlVal (multPanel, MULTPNL_31HITNUM, stop31Events);
        SetCtrlVal (multPanel, MULTPNL_32HITNUM, stop32Events);
        SetCtrlVal (multPanel, MULTPNL_33HITNUM, stop33Events);
        SetCtrlVal (multPanel, MULTPNL_34HITNUM, stop34Events);
```

```
                    SetCtrlVal (multPanel, MULTPNL_35HITNUM, stop35Events);
                    SetCtrlVal (multPanel, MULTPNL_36HITNUM, stop36Events);
                    SetCtrlVal (multPanel, MULTPNL_37HITNUM, stop37Events);
                    SetCtrlVal (multPanel, MULTPNL_38HITNUM, stop38Events);
                    SetCtrlVal (multPanel, MULTPNL_39HITNUM, stop39Events);
                    SetCtrlVal (multPanel, MULTPNL_40HITNUM, stop40Events);
                    SetCtrlVal (multPanel, MULTPNL_41HITNUM, stop41Events);
                    SetCtrlVal (multPanel, MULTPNL_42HITNUM, stop42Events);
                    SetCtrlVal (multPanel, MULTPNL_43HITNUM, stop43Events);
                    SetCtrlVal (multPanel, MULTPNL_44HITNUM, stop44Events);
                    SetCtrlVal (multPanel, MULTPNL_45HITNUM, stop45Events);
                    SetCtrlVal (multPanel, MULTPNL_46HITNUM, stop46Events);
                    SetCtrlVal (multPanel, MULTPNL_47HITNUM, stop47Events);
                    SetCtrlVal (multPanel, MULTPNL_48HITNUM, stop48Events);
                    SetCtrlVal (multPanel, MULTPNL_49HITNUM, stop49Events);
                    SetCtrlVal (multPanel, MULTPNL_50HITNUM, stop50Events);

        d=2;
                    endtime=0;
                    det=0;
                    counter=0;
                    f = 0;
                    t1=0;
                    t2=0;


if(stop1Events>0){
histAllocateEA1(stop1Events);SetCtrlAttribute (cnspecPnl, COINPNL_AR1_SW, ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES1, ATTR_DIMMED, 0);}
if(stop2Events>0){
histAllocateEA2(stop2Events*2);SetCtrlAttribute (cnspecPnl, COINPNL_AR2_SW, ATTR_DIMMED,
0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES2, ATTR_DIMMED, 0);}
if(stop3Events>0){
histAllocateEA3(stop3Events*3);SetCtrlAttribute (cnspecPnl, COINPNL_AR3_SW, ATTR_DIMMED,
0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES3, ATTR_DIMMED, 0);}
if(stop4Events>0){
histAllocateEA4(stop4Events*4);SetCtrlAttribute (cnspecPnl, COINPNL_AR4_SW, ATTR_DIMMED,
0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES4, ATTR_DIMMED, 0);}
if(stop5Events>0){
histAllocateEA5(stop5Events*5);SetCtrlAttribute (cnspecPnl, COINPNL_AR5_SW, ATTR_DIMMED,
0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES5, ATTR_DIMMED, 0);}
if(stop6Events>0){
histAllocateEA6(stop6Events*6);SetCtrlAttribute (cnspecPnl, COINPNL_AR6_SW, ATTR_DIMMED,
0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES6, ATTR_DIMMED, 0);}
if(stop7Events>0){
histAllocateEA7(stop7Events*7);SetCtrlAttribute (cnspecPnl, COINPNL_AR7_SW, ATTR_DIMMED,
0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES7, ATTR_DIMMED, 0);}
```

```
if(stop8Events>0){
histAllocateEA8(stop8Events*8);SetCtrlAttribute (cnspecPnl, COINPNL_AR8_SW, ATTR_DIMMED,
0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES8, ATTR_DIMMED, 0);}
if(stop9Events>0){
histAllocateEA9(stop9Events*9);SetCtrlAttribute (cnspecPnl, COINPNL_AR9_SW, ATTR_DIMMED,
0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES9, ATTR_DIMMED, 0);}
if(stop10Events>0){
histAllocateEA10(stop10Events*10);SetCtrlAttribute (cnspecPnl, COINPNL_AR10_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES10, ATTR_DIMMED, 0);}
if(stop11Events>0){
histAllocateEA11(stop11Events*11);SetCtrlAttribute (cnspecPnl, COINPNL_AR11_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES11, ATTR_DIMMED, 0);}
if(stop12Events>0){
histAllocateEA12(stop12Events*12);SetCtrlAttribute (cnspecPnl, COINPNL_AR12_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES12, ATTR_DIMMED, 0);}
if(stop13Events>0){
histAllocateEA13(stop13Events*13);SetCtrlAttribute (cnspecPnl, COINPNL_AR13_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES13, ATTR_DIMMED, 0);}
if(stop14Events>0){
histAllocateEA14(stop14Events*14);SetCtrlAttribute (cnspecPnl, COINPNL_AR14_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES14, ATTR_DIMMED, 0);}
if(stop15Events>0){
histAllocateEA15(stop15Events*15);SetCtrlAttribute (cnspecPnl, COINPNL_AR15_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES15, ATTR_DIMMED, 0);}
if(stop16Events>0){
histAllocateEA16(stop16Events*16);SetCtrlAttribute (cnspecPnl, COINPNL_AR16_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES16, ATTR_DIMMED, 0);}
if(stop17Events>0){
histAllocateEA17(stop17Events*17);SetCtrlAttribute (cnspecPnl, COINPNL_AR17_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES17, ATTR_DIMMED, 0);}
if(stop18Events>0){
histAllocateEA18(stop18Events*18);SetCtrlAttribute (cnspecPnl, COINPNL_AR18_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES18, ATTR_DIMMED, 0);}
if(stop19Events>0){
histAllocateEA19(stop19Events*19);SetCtrlAttribute (cnspecPnl, COINPNL_AR19_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES19, ATTR_DIMMED, 0);}
if(stop20Events>0){
histAllocateEA20(stop20Events*20);SetCtrlAttribute (cnspecPnl, COINPNL_AR20_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES20, ATTR_DIMMED, 0);}
if(stop21Events>0){
```

```
histAllocateEA21(stop21Events*21);SetCtrlAttribute (cnspecPnl, COINPNL_AR21_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES21, ATTR_DIMMED, 0);}
if(stop22Events>0){
histAllocateEA22(stop22Events*22);SetCtrlAttribute (cnspecPnl, COINPNL_AR22_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES22, ATTR_DIMMED, 0);}
if(stop23Events>0){
histAllocateEA23(stop23Events*23);SetCtrlAttribute (cnspecPnl, COINPNL_AR23_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES23, ATTR_DIMMED, 0);}
if(stop24Events>0){
histAllocateEA24(stop24Events*24);SetCtrlAttribute (cnspecPnl, COINPNL_AR24_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES24, ATTR_DIMMED, 0);}
if(stop25Events>0){
histAllocateEA25(stop25Events*25);SetCtrlAttribute (cnspecPnl, COINPNL_AR25_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES25, ATTR_DIMMED, 0);}
if(stop26Events>0){
histAllocateEA26(stop26Events*26);SetCtrlAttribute (cnspecPnl, COINPNL_AR26_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES26, ATTR_DIMMED, 0);}
if(stop27Events>0){
histAllocateEA27(stop27Events*27);SetCtrlAttribute (cnspecPnl, COINPNL_AR27_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES27, ATTR_DIMMED, 0);}
if(stop28Events>0){
histAllocateEA28(stop28Events*28);SetCtrlAttribute (cnspecPnl, COINPNL_AR28_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES28, ATTR_DIMMED, 0);}
if(stop29Events>0){
histAllocateEA29(stop29Events*29);SetCtrlAttribute (cnspecPnl, COINPNL_AR29_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES29, ATTR_DIMMED, 0);}
if(stop30Events>0){
histAllocateEA30(stop30Events*30);SetCtrlAttribute (cnspecPnl, COINPNL_AR30_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES30, ATTR_DIMMED, 0);}
if(stop31Events>0){
histAllocateEA31(stop31Events*31);SetCtrlAttribute (cnspecPnl, COINPNL_AR31_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES31, ATTR_DIMMED, 0);}
if(stop32Events>0){
histAllocateEA32(stop32Events*32);SetCtrlAttribute (cnspecPnl, COINPNL_AR32_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES32, ATTR_DIMMED, 0);}
if(stop33Events>0){
histAllocateEA33(stop33Events*33);SetCtrlAttribute (cnspecPnl, COINPNL_AR33_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES33, ATTR_DIMMED, 0);}
if(stop34Events>0){
```

```
histAllocateEA34(stop34Events*34);SetCtrlAttribute (cnspecPnl, COINPNL_AR34_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES34, ATTR_DIMMED, 0);}
if(stop35Events>0){
histAllocateEA35(stop35Events*35);SetCtrlAttribute (cnspecPnl, COINPNL_AR35_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES35, ATTR_DIMMED, 0);}
if(stop36Events>0){
histAllocateEA36(stop36Events*36);SetCtrlAttribute (cnspecPnl, COINPNL_AR36_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES36, ATTR_DIMMED, 0);}
if(stop37Events>0){
histAllocateEA37(stop37Events*37);SetCtrlAttribute (cnspecPnl, COINPNL_AR37_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES37, ATTR_DIMMED, 0);}
if(stop38Events>0){
histAllocateEA38(stop38Events*38);SetCtrlAttribute (cnspecPnl, COINPNL_AR38_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES38, ATTR_DIMMED, 0);}
if(stop39Events>0){
histAllocateEA39(stop39Events*39);SetCtrlAttribute (cnspecPnl, COINPNL_AR39_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES39, ATTR_DIMMED, 0);}
if(stop40Events>0){
histAllocateEA40(stop40Events*40);SetCtrlAttribute (cnspecPnl, COINPNL_AR40_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES40, ATTR_DIMMED, 0);}
if(stop41Events>0){
histAllocateEA41(stop41Events*41);SetCtrlAttribute (cnspecPnl, COINPNL_AR41_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES41, ATTR_DIMMED, 0);}
if(stop42Events>0){
histAllocateEA42(stop42Events*42);SetCtrlAttribute (cnspecPnl, COINPNL_AR42_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES42, ATTR_DIMMED, 0);}
if(stop43Events>0){
histAllocateEA43(stop43Events*43);SetCtrlAttribute (cnspecPnl, COINPNL_AR43_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES43, ATTR_DIMMED, 0);}
if(stop44Events>0){
histAllocateEA44(stop44Events*44);SetCtrlAttribute (cnspecPnl, COINPNL_AR44_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES44, ATTR_DIMMED, 0);}
if(stop45Events>0){
histAllocateEA45(stop45Events*45);SetCtrlAttribute (cnspecPnl, COINPNL_AR45_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES45, ATTR_DIMMED, 0);}
if(stop46Events>0){
histAllocateEA46(stop46Events*46);SetCtrlAttribute (cnspecPnl, COINPNL_AR46_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES46, ATTR_DIMMED, 0);}
if(stop47Events>0){
```

```
histAllocateEA47(stop47Events*47);SetCtrlAttribute (cnspecPnl, COINPNL_AR47_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES47, ATTR_DIMMED, 0);}
if(stop48Events>0){
histAllocateEA48(stop48Events*48);SetCtrlAttribute (cnspecPnl, COINPNL_AR48_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES48, ATTR_DIMMED, 0);}
if(stop49Events>0){
histAllocateEA49(stop49Events*49);SetCtrlAttribute (cnspecPnl, COINPNL_AR49_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES49, ATTR_DIMMED, 0);}
if(stop50Events>0){
histAllocateEA50(stop50Events*50);SetCtrlAttribute (cnspecPnl, COINPNL_AR50_SW,
ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_SAVRES50, ATTR_DIMMED, 0);}

if(ch0>0){
SetCtrlAttribute (cnspecPnl, COINPNL_CH0_SW, ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_CH0_SWEN, ATTR_DIMMED, 0);}
if(ch1>0){
SetCtrlAttribute (cnspecPnl, COINPNL_CH1_SW, ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_CH1_SWEN, ATTR_DIMMED, 0);}
if(ch2>0){
SetCtrlAttribute (cnspecPnl, COINPNL_CH2_SW, ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_CH2_SWEN, ATTR_DIMMED, 0);}
if(ch3>0){
SetCtrlAttribute (cnspecPnl, COINPNL_CH3_SW, ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_CH3_SWEN, ATTR_DIMMED, 0);}
if(ch4>0){
SetCtrlAttribute (cnspecPnl, COINPNL_CH4_SW, ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_CH4_SWEN, ATTR_DIMMED, 0);}
if(ch5>0){
SetCtrlAttribute (cnspecPnl, COINPNL_CH5_SW, ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_CH5_SWEN, ATTR_DIMMED, 0);}
if(ch6>0){
SetCtrlAttribute (cnspecPnl, COINPNL_CH6_SW, ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_CH6_SWEN, ATTR_DIMMED, 0);}
if(ch7>0){
SetCtrlAttribute (cnspecPnl, COINPNL_CH7_SW, ATTR_DIMMED, 0);
SetCtrlAttribute (cnspecPnl, COINPNL_CH7_SWEN, ATTR_DIMMED, 0);}


                d=((size/2)-1);
                top1:
                while (d>=0){

                                        while((d>=0)&& ((array[d]&61440) != 32768)){

                                                                det =
array[d]&61440;
                                                                t1 =
array[d]&4095;
                                                                --d;
```

```
                                                                switch (det)
                                                                        {
                                                                        case 0 :
        ch0+=1;
        break;
                                                                        case
4096 :
        ch1++;
        break;
                                                                        case
8192 :
        ch2++;
        break;
                                                                        case
12288 :
        ch3++;
        break;
                                                                        case
16384 :
        ch4++;
        break;
                                                                        case
20480 :
        ch5++;
        break;
                                                                        case
24576 :
        ch6++;
        break;
                                                                        case
28672 :
        ch7++;
        break;
                                                                        default:
        --d;
```

```
counter = 0;

goto top1;

break;

}


                                                                endtime =
array[d]+(t1*32768);
                                                                counter++;
                                                                --d;
                                                                f=counter;

                                                                tempT[f-1] =
endtime;
                                                                tempD[f-1] =
det;


                        }//while(Aarray[d]&64512 != 51200)

                                                switch (counter)
                                                        {

                                                        case 0:--d;--d;break;
                                                        case 1:for(x=0;x<=(counter-1);x++){

        Event1TArray[ea1]=tempT[x];Event1DArray[ea1]=tempD[x];ea1++;}
                                                        counter = 0;x=0;break;
                                                        case 2:for(x=0;x<=(counter-1);x++){

        Event2TArray[ea2]=tempT[x];Event2DArray[ea2]=tempD[x];ea2++;}
                                                        counter = 0;x=0;break;
                                                        case 3:for(x=0;x<=(counter-1);x++){

        Event3TArray[ea3]=tempT[x];Event3DArray[ea3]=tempD[x];ea3++;}
                                                        counter = 0;x=0;break;
                                                        case 4:for(x=0;x<=(counter-1);x++){

        Event4TArray[ea4]=tempT[x];Event4DArray[ea4]=tempD[x];ea4++;}
                                                        counter = 0;x=0;break;
                                                        case 5:for(x=0;x<=(counter-1);x++){

        Event5TArray[ea5]=tempT[x];Event5DArray[ea5]=tempD[x];ea5++;}
                                                        counter = 0;x=0;break;
                                                        case 6:for(x=0;x<=(counter-1);x++){

        Event6TArray[ea6]=tempT[x];Event6DArray[ea6]=tempD[x];ea6++;}
```

```
                                                    counter = 0;x=0;break;
                                                    case 7:for(x=0;x<=(counter-1);x++){

Event7TArray[ea7]=tempT[x];Event7DArray[ea7]=tempD[x];ea7++;}
                                                    counter = 0;x=0;break;
                                                    case 8:for(x=0;x<=(counter-1);x++){

Event8TArray[ea8]=tempT[x];Event8DArray[ea8]=tempD[x];ea8++;}
                                                    counter = 0;x=0;break;
                                                    case 9:for(x=0;x<=(counter-1);x++){

Event9TArray[ea9]=tempT[x];Event9DArray[ea9]=tempD[x];ea9++;}
                                                    counter = 0;x=0;break;
                                                    case 10:for(x=0;x<=(counter-1);x++){

Event10TArray[ea10]=tempT[x];Event10DArray[ea10]=tempD[x];ea10++;}
                                                    counter = 0;x=0;break;
                                                    case 11:for(x=0;x<=(counter-1);x++){

Event11TArray[ea11]=tempT[x];Event11DArray[ea11]=tempD[x];ea11++;}
                                                    counter = 0;x=0;break;
                                                    case 12:for(x=0;x<=(counter-1);x++){

Event12TArray[ea12]=tempT[x];Event12DArray[ea12]=tempD[x];ea12++;}
                                                    counter = 0;x=0;break;
                                                    case 13:for(x=0;x<=(counter-1);x++){

Event13TArray[ea13]=tempT[x];Event13DArray[ea13]=tempD[x];ea13++;}
                                                    counter = 0;x=0;break;
                                                    case 14:for(x=0;x<=(counter-1);x++){

Event14TArray[ea14]=tempT[x];Event14DArray[ea14]=tempD[x];ea14++;}
                                                    counter = 0;x=0;break;
                                                    case 15:for(x=0;x<=(counter-1);x++){

Event15TArray[ea15]=tempT[x];Event15DArray[ea15]=tempD[x];ea15++;}
                                                    counter = 0;x=0;break;
                                                    case 16:for(x=0;x<=(counter-1);x++){

Event16TArray[ea16]=tempT[x];Event16DArray[ea16]=tempD[x];ea16++;}
                                                    counter = 0;x=0;break;
                                                    case 17:for(x=0;x<=(counter-1);x++){

Event17TArray[ea17]=tempT[x];Event17DArray[ea17]=tempD[x];ea17++;}
                                                    counter = 0;x=0;break;
                                                    case 18:for(x=0;x<=(counter-1);x++){

Event18TArray[ea18]=tempT[x];Event18DArray[ea18]=tempD[x];ea18++;}
                                                    counter = 0;x=0;break;
                                                    case 19:for(x=0;x<=(counter-1);x++){

Event19TArray[ea19]=tempT[x];Event19DArray[ea19]=tempD[x];ea19++;}
                                                    counter = 0;x=0;break;
```

```
                                                    case 20:for(x=0;x<=(counter-1);x++){

Event20TArray[ea20]=tempT[x];Event20DArray[ea20]=tempD[x];ea20++;}
                                                    counter = 0;x=0;break;
                                                    case 21:for(x=0;x<=(counter-1);x++){

Event21TArray[ea21]=tempT[x];Event21DArray[ea21]=tempD[x];ea21++;}
                                                    counter = 0;x=0;break;
                                                    case 22:for(x=0;x<=(counter-1);x++){

Event22TArray[ea22]=tempT[x];Event22DArray[ea22]=tempD[x];ea22++;}
                                                    counter = 0;x=0;break;
                                                    case 23:for(x=0;x<=(counter-1);x++){

Event23TArray[ea23]=tempT[x];Event23DArray[ea23]=tempD[x];ea23++;}
                                                    counter = 0;x=0;break;
                                                    case 24:for(x=0;x<=(counter-1);x++){

Event24TArray[ea24]=tempT[x];Event24DArray[ea24]=tempD[x];ea24++;}
                                                    counter = 0;x=0;break;
                                                    case 25:for(x=0;x<=(counter-1);x++){

Event25TArray[ea25]=tempT[x];Event25DArray[ea25]=tempD[x];ea25++;}
                                                    counter = 0;x=0;break;
                                                    case 26:for(x=0;x<=(counter-1);x++){

Event26TArray[ea26]=tempT[x];Event26DArray[ea26]=tempD[x];ea26++;}
                                                    counter = 0;x=0;break;
                                                    case 27:for(x=0;x<=(counter-1);x++){

Event27TArray[ea27]=tempT[x];Event27DArray[ea27]=tempD[x];ea27++;}
                                                    counter = 0;x=0;break;
                                                    case 28:for(x=0;x<=(counter-1);x++){

Event28TArray[ea28]=tempT[x];Event28DArray[ea28]=tempD[x];ea28++;}
                                                    counter = 0;x=0;break;
                                                    case 29:for(x=0;x<=(counter-1);x++){

Event29TArray[ea29]=tempT[x];Event29DArray[ea29]=tempD[x];ea29++;}
                                                    counter = 0;x=0;break;
                                                    case 30:for(x=0;x<=(counter-1);x++){

Event30TArray[ea30]=tempT[x];Event30DArray[ea30]=tempD[x];ea30++;}
                                                    counter = 0;x=0;break;
                                                    case 31:for(x=0;x<=(counter-1);x++){

Event31TArray[ea31]=tempT[x];Event31DArray[ea31]=tempD[x];ea31++;}
                                                    counter = 0;x=0;break;
                                                    case 32:for(x=0;x<=(counter-1);x++){

Event32TArray[ea32]=tempT[x];Event32DArray[ea32]=tempD[x];ea32++;}
                                                    counter = 0;x=0;break;
                                                    case 33:for(x=0;x<=(counter-1);x++){
```

```
Event33TArray[ea33]=tempT[x];Event33DArray[ea33]=tempD[x];ea33++;}
                                                    counter = 0;x=0;break;
                                                    case 34:for(x=0;x<=(counter-1);x++){

Event34TArray[ea34]=tempT[x];Event34DArray[ea34]=tempD[x];ea34++;}
                                                    counter = 0;x=0;break;
                                                    case 35:for(x=0;x<=(counter-1);x++){

Event35TArray[ea35]=tempT[x];Event35DArray[ea35]=tempD[x];ea35++;}
                                                    counter = 0;x=0;break;
                                                    case 36:for(x=0;x<=(counter-1);x++){

Event36TArray[ea36]=tempT[x];Event36DArray[ea36]=tempD[x];ea36++;}
                                                    counter = 0;x=0;break;
                                                    case 37:for(x=0;x<=(counter-1);x++){

Event37TArray[ea37]=tempT[x];Event37DArray[ea37]=tempD[x];ea37++;}
                                                    counter = 0;x=0;break;
                                                    case 38:for(x=0;x<=(counter-1);x++){

Event38TArray[ea38]=tempT[x];Event38DArray[ea38]=tempD[x];ea38++;}
                                                    counter = 0;x=0;break;
                                                    case 39:for(x=0;x<=(counter-1);x++){

Event39TArray[ea39]=tempT[x];Event39DArray[ea39]=tempD[x];ea39++;}
                                                    counter = 0;x=0;break;
                                                    case 40:for(x=0;x<=(counter-1);x++){

Event40TArray[ea40]=tempT[x];Event40DArray[ea40]=tempD[x];ea40++;}
                                                    counter = 0;x=0;break;
                                                    case 41:for(x=0;x<=(counter-1);x++){

Event41TArray[ea41]=tempT[x];Event41DArray[ea41]=tempD[x];ea41++;}
                                                    counter = 0;x=0;break;
                                                    case 42:for(x=0;x<=(counter-1);x++){

Event42TArray[ea42]=tempT[x];Event42DArray[ea42]=tempD[x];ea42++;}
                                                    counter = 0;x=0;break;
                                                    case 43:for(x=0;x<=(counter-1);x++){

Event43TArray[ea43]=tempT[x];Event43DArray[ea43]=tempD[x];ea43++;}
                                                    counter = 0;x=0;break;
                                                    case 44:for(x=0;x<=(counter-1);x++){

Event44TArray[ea44]=tempT[x];Event44DArray[ea44]=tempD[x];ea44++;}
                                                    counter = 0;x=0;break;
                                                    case 45:for(x=0;x<=(counter-1);x++){

Event45TArray[ea45]=tempT[x];Event45DArray[ea45]=tempD[x];ea45++;}
                                                    counter = 0;x=0;break;
                                                    case 46:for(x=0;x<=(counter-1);x++){
```

```
        Event46TArray[ea46]=tempT[x];Event46DArray[ea46]=tempD[x];ea46++;}
                                                        counter = 0;x=0;break;
                                                        case 47:for(x=0;x<=(counter-1);x++){

        Event47TArray[ea47]=tempT[x];Event47DArray[ea47]=tempD[x];ea47++;}
                                                        counter = 0;x=0;break;
                                                        case 48:for(x=0;x<=(counter-1);x++){

        Event48TArray[ea48]=tempT[x];Event48DArray[ea48]=tempD[x];ea48++;}
                                                        counter = 0;x=0;break;
                                                        case 49:for(x=0;x<=(counter-1);x++){

        Event49TArray[ea49]=tempT[x];Event49DArray[ea49]=tempD[x];ea49++;}
                                                        counter = 0;x=0;break;
                                                        case 50:for(x=0;x<=(counter-1);x++){

        Event50TArray[ea50]=tempT[x];Event50DArray[ea50]=tempD[x];ea50++;}
                                                        counter = 0;x=0;break;

                                                        default:counter=0;break;
                                                        } // switch(counter)


          }//while (array[d] != 0 )
                SetCtrlVal (mnPanel, MN_PANEL_NUMEVENT, totalEvents);



}

}
/*********************************Data Acqisition
Functions*****************************/
void CVICALLBACK MenuStartDataAcquireCB (int menuBar, int menuItem, void *callbackData,
                int panel)
{
        DisplayPanel(expPnl);
}

void CVICALLBACK MenuStopACB (int menuBar, int menuItem, void *callbackData,
                int panel)
{
                r = iLoopCount;
                iHalfBufsToRead = r;
                SetCtrlVal (expPnl, EXP_PNL_TM2TXTBOX, TimeStr());
                SetCtrlVal (expPnl, EXP_PNL_DT2TXTBOX, DateStr());
                DisplayPanel(expPnl);
                GetProjectDir (proj_dir);

                GetCtrlVal (expPnl, EXP_PNL_TM1TXTBOX, Time);
                GetCtrlVal (expPnl, EXP_PNL_DT1TXTBOX, Date);
                GetCtrlVal (expPnl, EXP_PNL_TM2TXTBOX, Time2);
```

```
                GetCtrlVal (expPnl, EXP_PNL_DT2TXTBOX, Date2);
                GetCtrlVal (expPnl, EXP_PNL_OPTEXTBOX, operator);
                GetCtrlVal (expPnl, EXP_PNL_IDTEXTBOX, id);
                GetCtrlVal (expPnl, EXP_PNL_TRTEXTBOX, target);
                GetCtrlVal (expPnl, EXP_PNL_PITEXTBOX, pidat);
                GetCtrlVal (expPnl, EXP_PNL_SIKTXTBOX, sidat);
                GetCtrlVal (expPnl, EXP_PNL_STDTXTBOX, startdat);
                GetCtrlVal (expPnl, EXP_PNL_ST1TXTBOX, stop1dat);
                GetCtrlVal (expPnl, EXP_PNL_ST2TXTBOX, stop2dat);
                GetCtrlVal (expPnl, EXP_PNL_MAXCHAN, &chanmax);


        FileSelectPopup (proj_dir, "*.exp", ".exp",
                                                "Select Experimental Parameter File",
                                                VAL_SAVE_BUTTON, 0, 1, 1, 1, file_name);

                WriteFileHandle = OpenFile (file_name, VAL_WRITE_ONLY, VAL_TRUNCATE,
                                                                        VAL_BINARY);

                FmtFile (WriteFileHandle,"%s\n %s\n %s\n %s\n %s\n %s\n %s\n %s\n %s\n %s\n %s\n
%s\n %d\n",
                id, Date, Time, Date2, Time2, operator, target,pidat, sidat, startdat, stop1dat, stop2dat,
chanmax);

}
int CVICALLBACK StartAcquire8CB (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event)
                {

                case EVENT_COMMIT:
                                                        StartDataAcquire8CB();
                                                        HidePanel(expPnl);

                        break;
                }
        return 0;
}
int CVICALLBACK StartAcquireACB (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event)
                {
                case EVENT_COMMIT:
                                                        StartDataAcquireA();
                                                        HidePanel(expPnl);
                        break;
                }
        return 0;
}
int CVICALLBACK StartAcquire6CB (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
```

```
{
	switch (event)
		{
		case EVENT_COMMIT:
								StartDataAcquire6CB();
								HidePanel(expPnl);
			break;
		}
	return 0;
}
int CVICALLBACK HideAcquireCB (int panel, int control, int event,
		void *callbackData, int eventData1, int eventData2)
{
	switch (event)
		{
		case EVENT_COMMIT:
							HidePanel(expPnl);
			break;
		}
	return 0;
}
void CVICALLBACK MenuDisplayExpParmCB (int menuBar, int menuItem, void *callbackData,
		int panel)
{
	DisplayPanel(expPnl);
}
/***************************************************Graphing*************************
****/
int CVICALLBACK GraphCB (int panel, int control, int event,
		void *callbackData, int eventData1, int eventData2)
{
	switch (event)
		{

		case EVENT_COMMIT:

			break;
		case EVENT_LEFT_CLICK:

		SetCursorAttribute (mnPanel, MN_PANEL_GRAPH, 2, ATTR_CURSOR_COLOR,
								VAL_RED);
		SetCursorAttribute (mnPanel, MN_PANEL_GRAPH, 1, ATTR_CURSOR_COLOR,
								VAL_BLUE);
		SetCursorAttribute (mnPanel, MN_PANEL_GRAPH, 3, ATTR_CURSOR_COLOR,
								VAL_MAGENTA);
		GetGraphCursor (mnPanel, MN_PANEL_GRAPH, 3, &x1ref, &y1ref);

		SetCtrlVal (mnPanel, MN_PANEL_NUMXREF, x1ref);
		SetCtrlVal (mnPanel, MN_PANEL_NUMYREF, y1ref);
			break;
		case EVENT_LEFT_DOUBLE_CLICK:

			SetAxisScalingMode (mnPanel, MN_PANEL_GRAPH, VAL_LEFT_YAXIS,
```

```
                                                                    VAL_AUTOSCALE, y1, y2);
                        SetAxisScalingMode (mnPanel, MN_PANEL_GRAPH, VAL_XAXIS,
VAL_AUTOSCALE,
                                                                    y1, y2);
                        break;
                case EVENT_RIGHT_CLICK:

                        break;
                }
        return 0;
}
void CVICALLBACK MenuCh0CB (int menuBar, int menuItem, void *callbackData,
                int panel)
{
        DeleteGraphPlot (mnPanel, MN_PANEL_GRAPH, -1, VAL_IMMEDIATE_DRAW);
        RefreshGraph (mnPanel, MN_PANEL_GRAPH);
        currHist=coinspec0;
        PlotY (mnPanel, MN_PANEL_GRAPH, currHist, D, VAL_INTEGER,
                        VAL_THIN_LINE, VAL_NO_POINT, VAL_SOLID, 1, VAL_BLACK);

}

void CVICALLBACK MenuCh1CB (int menuBar, int menuItem, void *callbackData,
                int panel)
{
        DeleteGraphPlot (mnPanel, MN_PANEL_GRAPH, -1, VAL_IMMEDIATE_DRAW);
        RefreshGraph (mnPanel, MN_PANEL_GRAPH);
        currHist=coinspec1;
        PlotY (mnPanel, MN_PANEL_GRAPH, currHist, D, VAL_INTEGER,
                        VAL_THIN_LINE, VAL_NO_POINT, VAL_SOLID, 1, VAL_BLACK);

}

void CVICALLBACK MenuCh2CB (int menuBar, int menuItem, void *callbackData,
                int panel)
{
        DeleteGraphPlot (mnPanel, MN_PANEL_GRAPH, -1, VAL_IMMEDIATE_DRAW);
        RefreshGraph (mnPanel, MN_PANEL_GRAPH);
        currHist=coinspec2;
        PlotY (mnPanel, MN_PANEL_GRAPH, currHist, D, VAL_INTEGER,
                        VAL_THIN_LINE, VAL_NO_POINT, VAL_SOLID, 1, VAL_BLACK);

}

void CVICALLBACK MenuCh3CB (int menuBar, int menuItem, void *callbackData,
                int panel)
{
        DeleteGraphPlot (mnPanel, MN_PANEL_GRAPH, -1, VAL_IMMEDIATE_DRAW);
        RefreshGraph (mnPanel, MN_PANEL_GRAPH);
        currHist=coinspec3;
        PlotY (mnPanel, MN_PANEL_GRAPH, currHist, D, VAL_INTEGER,
                        VAL_THIN_LINE, VAL_NO_POINT, VAL_SOLID, 1, VAL_BLACK);
```

```
}

void CVICALLBACK MenuCh4CB (int menuBar, int menuItem, void *callbackData,
		int panel)
{
	DeleteGraphPlot (mnPanel, MN_PANEL_GRAPH, -1, VAL_IMMEDIATE_DRAW);
	RefreshGraph (mnPanel, MN_PANEL_GRAPH);
	currHist=coinspec4;
	PlotY (mnPanel, MN_PANEL_GRAPH, currHist, D, VAL_INTEGER,
			VAL_THIN_LINE, VAL_NO_POINT, VAL_SOLID, 1, VAL_BLACK);

}

void CVICALLBACK MenuCh5CB (int menuBar, int menuItem, void *callbackData,
		int panel)
{
	DeleteGraphPlot (mnPanel, MN_PANEL_GRAPH, -1, VAL_IMMEDIATE_DRAW);
	RefreshGraph (mnPanel, MN_PANEL_GRAPH);
	currHist=coinspec5;
	PlotY (mnPanel, MN_PANEL_GRAPH, currHist, D, VAL_INTEGER,
			VAL_THIN_LINE, VAL_NO_POINT, VAL_SOLID, 1, VAL_BLACK);

}

void CVICALLBACK MenuCh6CB (int menuBar, int menuItem, void *callbackData,
		int panel)
{
	DeleteGraphPlot (mnPanel, MN_PANEL_GRAPH, -1, VAL_IMMEDIATE_DRAW);
	RefreshGraph (mnPanel, MN_PANEL_GRAPH);
	currHist=coinspec6;
	PlotY (mnPanel, MN_PANEL_GRAPH, currHist, D, VAL_INTEGER,
			VAL_THIN_LINE, VAL_NO_POINT, VAL_SOLID, 1, VAL_BLACK);

}

void CVICALLBACK MenuCh7CB (int menuBar, int menuItem, void *callbackData,
		int panel)
{
	DeleteGraphPlot (mnPanel, MN_PANEL_GRAPH, -1, VAL_IMMEDIATE_DRAW);
	RefreshGraph (mnPanel, MN_PANEL_GRAPH);
	currHist=coinspec7;
	PlotY (mnPanel, MN_PANEL_GRAPH, currHist, D, VAL_INTEGER,
			VAL_THIN_LINE, VAL_NO_POINT, VAL_SOLID, 1, VAL_BLACK);

}

void CVICALLBACK MenuInsertFileNameCB (int menuBar, int menuItem, void *callbackData,
		int panel)
{
		GetGraphCursor (mnPanel, MN_PANEL_GRAPH, 1, &x1, &y1);
		PlotText (mnPanel, MN_PANEL_GRAPH, x1, y1, file_name,
VAL_APP_META_FONT,
				VAL_BLACK, VAL_TRANSPARENT);
```

```
}
void CVICALLBACK MenuInsertChannelCB (int menuBar, int menuItem, void *callbackData,
		int panel)
{
		GetGraphCursor (mnPanel, MN_PANEL_GRAPH, 3, &x1, &y1);
		response = ConfirmPopup ("m/z?",
                                                                "If (No) then channel number will
be used");
		if (response>0){
                        Fmt(buf,"%s<  m/z: %f[p0] - Intensity: %f[p6] ", x1, y1);
                        PlotText (mnPanel, MN_PANEL_GRAPH, x1, y1, buf,
VAL_APP_META_FONT,
                                                VAL_BLACK, VAL_TRANSPARENT);
                                                }
		else
                        Fmt(buf,"%s<  Channel: %f[p0] - Intensity: %f[p6] ", x1, y1);
                        PlotText (mnPanel, MN_PANEL_GRAPH, x1, y1, buf,
VAL_APP_META_FONT,
                                                VAL_BLACK, VAL_TRANSPARENT);

}
void CVICALLBACK menuInsertExtParCB (int menuBar, int menuItem, void *callbackData,
		int panel)
{

                GetGraphCursor (mnPanel, MN_PANEL_GRAPH, 1, &x1, &y1);
                PromptPopup ("", "Text to be inserted.", buf, 260);
                        PlotText (mnPanel, MN_PANEL_GRAPH, x1, y1, buf,
VAL_APP_META_FONT,
                                                VAL_BLACK, VAL_TRANSPARENT);


}
void CVICALLBACK MenuDateCB (int menuBar, int menuItem, void *callbackData,
		int panel)
{
	GetGraphCursor (mnPanel, MN_PANEL_GRAPH, 1, &x1, &y1);
	GetSystemDate (&mon, &dat, &yr);
	Fmt(buf,"%s< %d/%d/%d", mon, dat, yr);
	PlotText (mnPanel, MN_PANEL_GRAPH, x1, y1, buf, VAL_APP_META_FONT,
                                                VAL_BLACK, VAL_TRANSPARENT);

}
void CVICALLBACK MenuZoomAxisCB (int menuBar, int menuItem, void *callbackData,
		int panel)
{
	double x1, y1, x2, y2;
	GetGraphCursor (mnPanel, MN_PANEL_GRAPH, 1, &x1, &y1);
	GetGraphCursor (mnPanel, MN_PANEL_GRAPH, 2, &x2, &y2);
	if((x2>x1)&&(y2>y1)){
		SetAxisScalingMode (mnPanel, MN_PANEL_GRAPH, VAL_XAXIS,
VAL_MANUAL, x1, x2);
```

```
		SetAxisScalingMode (mnPanel, MN_PANEL_GRAPH, VAL_LEFT_YAXIS,
VAL_MANUAL, y1, y2);
	}

}
void CVICALLBACK MenuPeakAreaCB (int menuBar, int menuItem, void *callbackData,
		int panel)
{
		int x = 0;
		int counts = 0;
		GetGraphCursor (mnPanel, MN_PANEL_GRAPH, 1, &x1, &y1);
		GetGraphCursor (mnPanel, MN_PANEL_GRAPH, 2, &x2, &y2);
		if(x2>x1){
			for(x=x1;x<=x2;x++) {

					counts+=currHist[x];

				}
		SetCtrlVal (mnPanel, MN_PANEL_NUMPKAREA, counts);
				}
}
int CVICALLBACK LinearLogCB (int panel, int control, int event,
		void *callbackData, int eventData1, int eventData2)
{
	switch (event)
		{
		case EVENT_COMMIT:

		GetCtrlVal (mnPanel, MN_PANEL_LINLOG, &response);

		if (response==1){
				SetCtrlAttribute (mnPanel, MN_PANEL_GRAPH,
ATTR_YMAP_MODE, VAL_LOG);

				}

		if(response == 0){		SetCtrlAttribute (mnPanel, MN_PANEL_GRAPH,
ATTR_YMAP_MODE, VAL_LINEAR);
				}
			break;
		}
	return 0;
}
void CVICALLBACK MenuNormalizeCB (int menuBar, int menuItem, void *callbackData,
		int panel)
{
	DisplayPanel(normPanel);

}
int CVICALLBACK NormalizeCB (int panel, int control, int event,
		void *callbackData, int eventData1, int eventData2)
{
	switch (event)
```

```
                {

                case EVENT_COMMIT:
                normArr(D);

                for (x=0;x<=D-1;x++){

                        d=currHist[x];
                        dd=("%d", (int) d);
                        normArray[x] = d;

                                }

                        GetCtrlVal (normPanel, NORMPNL_NUMNORM, &ee);

                for (x=0;x<=D-1;x++){

                         dd=normArray[x];
                         normArray[x]=dd/ee;

                                }
                strcpy(buf,"Normalized Intensity");
                HidePanel(normPanel);
        DeleteGraphPlot (mnPanel, MN_PANEL_GRAPH, -1, VAL_IMMEDIATE_DRAW);
        RefreshGraph (mnPanel, MN_PANEL_GRAPH);
        SetCtrlAttribute (mnPanel, MN_PANEL_GRAPH, ATTR_YNAME, buf);
        SetCtrlAttribute (mnPanel, MN_PANEL_GRAPH, ATTR_YPRECISION, 4);
                SetCtrlAttribute (mnPanel, MN_PANEL_NUMYREF, ATTR_PRECISION, 4);
                PlotY (mnPanel, MN_PANEL_GRAPH, normArray, D, VAL_DOUBLE,
                                VAL_THIN_LINE, VAL_EMPTY_SQUARE, VAL_SOLID, 1,
VAL_BLACK);

                        break;
                }
        return 0;
}
void CVICALLBACK MenuCalibrateCB (int menuBar, int menuItem, void *callbackData,
                int panel)
{

                DisplayPanel(masscalibPnl);

}

int CVICALLBACK MassCalibrationCB (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event)
                {
                int i, x = 0;
                case EVENT_COMMIT:

                        MassArray(D);
```

```
                        GetCtrlVal (masscalibPnl, CALIBPNL_NUMCH1, &MassCalibY[0]);
                        GetCtrlVal (masscalibPnl, CALIBPNL_NUMCH2, &MassCalibY[1]);
                        GetCtrlVal (masscalibPnl, CALIBPNL_NUMCH3, &MassCalibY[2]);
                        GetCtrlVal (masscalibPnl, CALIBPNL_NUMCH4, &MassCalibY[3]);
                        GetCtrlVal (masscalibPnl, CALIBPNL_NUMCH5, &MassCalibY[4]);
                        GetCtrlVal (masscalibPnl, CALIBPNL_NUMCH6, &MassCalibY[5]);
                        GetCtrlVal (masscalibPnl, CALIBPNL_NUMCH7, &MassCalibY[6]);
                        GetCtrlVal (masscalibPnl, CALIBPNL_NUMCH8, &MassCalibY[7]);
                        GetCtrlVal (masscalibPnl, CALIBPNL_NUMMASS1, &MassCalibX[0]);
                        GetCtrlVal (masscalibPnl, CALIBPNL_NUMMASS2, &MassCalibX[1]);
                        GetCtrlVal (masscalibPnl, CALIBPNL_NUMMASS3, &MassCalibX[2]);
                        GetCtrlVal (masscalibPnl, CALIBPNL_NUMMASS4, &MassCalibX[3]);
                        GetCtrlVal (masscalibPnl, CALIBPNL_NUMMASS5, &MassCalibX[4]);
                        GetCtrlVal (masscalibPnl, CALIBPNL_NUMMASS6, &MassCalibX[5]);
                        GetCtrlVal (masscalibPnl, CALIBPNL_NUMMASS7, &MassCalibX[6]);
                        GetCtrlVal (masscalibPnl, CALIBPNL_NUMMASS8, &MassCalibX[7]);

                        i=0;
                        for (i=0;i<=7;i++){

                                ma=MassCalibX[i];
                                sm=sqrt(ma);
                                MassCalibX[i]=sm;}

                LinFit (MassCalibX, MassCalibY, 8, MassCalib, &slope, &intercept, &mse);

                SetCtrlVal (masscalibPnl, CALIBPNL_NUMSLOPE, slope);
                SetCtrlVal (masscalibPnl, CALIBPNL_NUMINTER, intercept);
                SetCtrlVal (masscalibPnl, CALIBPNL_NUMERR, mse);

                for (x = 0; x<=D-1; x++){

                        ca = ((x - intercept)/slope)*((x - intercept)/slope);

                        Mass[x] = ca;
                                                        }

        DeleteGraphPlot (mnPanel, MN_PANEL_GRAPH, -1, VAL_IMMEDIATE_DRAW);
        RefreshGraph (mnPanel, MN_PANEL_GRAPH);
        strcpy(buf,"m/z");
                SetCtrlAttribute (mnPanel, MN_PANEL_GRAPH, ATTR_XNAME, buf);
                SetCtrlAttribute (mnPanel, MN_PANEL_GRAPH, ATTR_XPRECISION, 2);
                PlotXY (mnPanel, MN_PANEL_GRAPH, Mass, currHist, D,
                                                        VAL_DOUBLE, VAL_INTEGER,
VAL_THIN_LINE,
                                                        VAL_EMPTY_SQUARE, VAL_SOLID,
1, VAL_BLACK);

                break;
                }
        return 0;
}
```

```
int CVICALLBACK NormalizeMassCB (int panel, int control, int event,
		void *callbackData, int eventData1, int eventData2)
{
	switch (event)
		{
		case EVENT_COMMIT:
		normArr(D);
		for (x=0;x<=D-1;x++){
			d=currHist[x];
			dd=("%d", (int) d);
			normArray[x] = d;

				}

			GetCtrlVal (normPanel, NORMPNL_NUMNORM, &ee);

		for (x=0;x<=D-1;x++){

			 dd=normArray[x];
			 normArray[x]=dd/ee;

				}
		HidePanel(normPanel);
	    DeleteGraphPlot (mnPanel, MN_PANEL_GRAPH, -1, VAL_IMMEDIATE_DRAW);
	    RefreshGraph (mnPanel, MN_PANEL_GRAPH);
	    strcpy(buf,"Normalized Intensity");
	    SetCtrlAttribute (mnPanel, MN_PANEL_GRAPH, ATTR_YNAME, buf);
	    SetCtrlAttribute (mnPanel, MN_PANEL_GRAPH, ATTR_XPRECISION, 2);
	    SetCtrlAttribute (mnPanel, MN_PANEL_GRAPH, ATTR_YPRECISION, 4);
		SetCtrlAttribute (mnPanel, MN_PANEL_NUMYREF, ATTR_PRECISION, 4);


		PlotXY (mnPanel, MN_PANEL_GRAPH, Mass, normArray, D, VAL_DOUBLE,
				VAL_DOUBLE, VAL_THIN_LINE, VAL_SOLID_SQUARE,
VAL_BLACK, 1,
				VAL_BLACK);


			break;
		}
	return 0;
}
/*****************************File/Printing/Copying Menu Functions***********************/
void CVICALLBACK MnuCopyGraphCB (int menuBar, int menuItem, void *callbackData,
		int panel)
{

	GetCtrlDisplayBitmap (mnPanel, MN_PANEL_GRAPH, 1, &copy);
	ClipboardPutBitmap (copy);

}
void CVICALLBACK MenuOpenFileCB (int menuBar, int menuItem, void *callbackData,
		int panel)
```

```
{

    if (FileSelectPopup ("", "*.cns", "*.cns", "Name of File to Read",
                                                        VAL_LOAD_BUTTON, 0, 1, 1, 1,
file_name) > 0){

            histCoinspec(D);
            FileToArray (file_name, coinspec, VAL_INTEGER, D, 1,
                                    VAL_GROUPS_TOGETHER,
VAL_GROUPS_AS_COLUMNS, VAL_ASCII);
            currHist = coinspec;
        DeleteGraphPlot (mnPanel, MN_PANEL_GRAPH, -1, VAL_IMMEDIATE_DRAW);
        RefreshGraph (mnPanel, MN_PANEL_GRAPH);
            PlotY (mnPanel, MN_PANEL_GRAPH, currHist, D, VAL_INTEGER,
                    VAL_THIN_LINE, VAL_EMPTY_SQUARE, VAL_SOLID, 1,
VAL_BLACK);


                                                        }
}
void CVICALLBACK MenuOpenCalibratedCB (int menuBar, int menuItem, void *callbackData,
            int panel)
{
            MessagePopup ("Attention",
                                    "Do not try to open a normalized mass spectrum");

            histCoinspec(D);
            MassArray(D);
    if (FileSelectPopup ("", "*.cmi", "*.cmi", "Name of File to Read",
                                                        VAL_LOAD_BUTTON, 0, 1, 1, 1,
file_name) > 0){
                    FileToArray (file_name, coinspec, VAL_INTEGER, D, 1,
                                                VAL_GROUPS_TOGETHER,
VAL_GROUPS_AS_COLUMNS, VAL_ASCII);

    if (FileSelectPopup ("", "*.cmm", "*.cmm", "Name of File to Read",
                                                        VAL_LOAD_BUTTON, 0, 1, 1, 1,
file_name) > 0){

                    FileToArray (file_name, Mass, VAL_DOUBLE, D, 1,
VAL_GROUPS_TOGETHER,
                                                VAL_GROUPS_AS_COLUMNS, VAL_ASCII);


                                                }

                                                }
            currHist = coinspec;
        DeleteGraphPlot (mnPanel, MN_PANEL_GRAPH, -1, VAL_IMMEDIATE_DRAW);
        RefreshGraph (mnPanel, MN_PANEL_GRAPH);
        strcpy(buf,"m/z");
            SetCtrlAttribute (mnPanel, MN_PANEL_GRAPH, ATTR_XNAME, buf);
            SetCtrlAttribute (mnPanel, MN_PANEL_GRAPH, ATTR_XPRECISION, 2);
```

```
		PlotXY (mnPanel, MN_PANEL_GRAPH, Mass, currHist, D,
                                                        VAL_DOUBLE, VAL_INTEGER,
VAL_THIN_LINE,
                                                        VAL_EMPTY_SQUARE, VAL_SOLID,
1, VAL_BLACK);


}
void CVICALLBACK MenuPrintCB (int menuBar, int menuItem, void *callbackData,
		int panel)
{

                        SetCursorAttribute (mnPanel, MN_PANEL_GRAPH, 1,
ATTR_CURSOR_COLOR,
                                                                VAL_TRANSPARENT);
                        SetCursorAttribute (mnPanel, MN_PANEL_GRAPH, 2,
ATTR_CURSOR_COLOR,
                                                                VAL_TRANSPARENT);
                        SetCursorAttribute (mnPanel, MN_PANEL_GRAPH, 3,
ATTR_CURSOR_COLOR,
                                                                VAL_TRANSPARENT);

	SetPrintAttribute (ATTR_PRINT_AREA_HEIGHT, VAL_USE_ENTIRE_PAPER);
	SetPrintAttribute (ATTR_PRINT_AREA_WIDTH, VAL_INTEGRAL_SCALE);
	PrintCtrl (mnPanel, MN_PANEL_GRAPH, "", 1, 1);

}
int CVICALLBACK CloseCalibrateCB (int panel, int control, int event,
		void *callbackData, int eventData1, int eventData2)
{
	switch (event)
		{
		case EVENT_COMMIT:HidePanel(masscalibPnl);

			break;
		}
	return 0;
}
void CVICALLBACK MenuSaveCB (int menuBar, int menuItem, void *callbackData,
		int panel)
{

	if (FileSelectPopup (proj_dir, "*.cns", "*.cns", "Name of File to Save",
                                                        VAL_SAVE_BUTTON, 0, 1, 1, 1, file_name) > 0)
{

                        ArrayToFile (file_name, currHist, VAL_UNSIGNED_INTEGER, D, 1,
                                                        VAL_GROUPS_TOGETHER,
VAL_GROUPS_AS_COLUMNS,
                                                        VAL_SEP_BY_TAB, 10, VAL_ASCII,
VAL_TRUNCATE);
                                                                }
}
```

```
void CVICALLBACK MenuSaveCalibratedCB (int menuBar, int menuItem, void *callbackData,
		int panel)
{
		MessagePopup ("Attention",
					"Do not try to save a normalized mass spectrum");

	if (FileSelectPopup (proj_dir, "*.cmm", "*.cmm", "Name of File to Save",
					VAL_OK_BUTTON, 0, 1, 1, 1, file_name) > 0) {

			ArrayToFile (file_name, Mass, VAL_DOUBLE, D, 1,
VAL_GROUPS_TOGETHER,
					VAL_GROUPS_AS_COLUMNS,
VAL_SEP_BY_TAB, 10, VAL_ASCII,
					VAL_TRUNCATE);

		if (FileSelectPopup (proj_dir, "*.cmi", "*.cmi", "Name of File to Save",
					VAL_OK_BUTTON, 0, 1, 1, 1, file_name) > 0) {


				ArrayToFile (file_name, currHist, VAL_INTEGER, D, 1,
					VAL_GROUPS_TOGETHER,
VAL_GROUPS_AS_COLUMNS,
					VAL_SEP_BY_TAB, 10,
VAL_ASCII, VAL_TRUNCATE);
					}
					}

}
/*****************************************Coincidence
Spectra**********************/
void CVICALLBACK MenuSingleParameterCB (int menuBar, int menuItem, void *callbackData,
		int panel)
{
    DisplayPanel(cnspecPnl);
}
int CVICALLBACK CalculateCoincidenceSpecCB (int panel, int control, int event,
		void *callbackData, int eventData1, int eventData2)
{

	switch (event)
		{
		case EVENT_COMMIT:

			GetCtrlVal (cnspecPnl, COINPNL_CH0_SWEN, &pen);
			if(pen>0){
					en=0;
				}
			GetCtrlVal (cnspecPnl, COINPNL_CH1_SWEN, &pen);
			if(pen>0){
					en=1;
				}
			GetCtrlVal (cnspecPnl, COINPNL_CH2_SWEN, &pen);
			if(pen>0){
```

```
                    en=2;
                }
GetCtrlVal (cnspecPnl, COINPNL_CH3_SWEN, &pen);
if(pen>0){
                    en=3;
                }
GetCtrlVal (cnspecPnl, COINPNL_CH4_SWEN, &pen);
if(pen>0){
                    en=4;
                }
GetCtrlVal (cnspecPnl, COINPNL_CH5_SWEN, &pen);
if(pen>0){
                    en=5;
                }
GetCtrlVal (cnspecPnl, COINPNL_CH6_SWEN, &pen);
if(pen>0){
                    en=6;
                }
GetCtrlVal (cnspecPnl, COINPNL_CH7_SWEN, &pen);
if(pen>0){
                    en=7;
                }

GetCtrlVal (cnspecPnl, COINPNL_NUMCOLO, &llm);
GetCtrlVal (multPanel, MULTPNL_10HITNUM, &hit10);
GetCtrlVal (cnspecPnl, COINPNL_NUMCOUP, &ulm);
GetCtrlVal (multPanel, MULTPNL_11HITNUM, &hit11);
GetCtrlVal (multPanel, MULTPNL_1HITNUM, &hit1);
GetCtrlVal (multPanel, MULTPNL_12HITNUM, &hit12);
GetCtrlVal (multPanel, MULTPNL_2HITNUM, &hit2);
GetCtrlVal (multPanel, MULTPNL_13HITNUM, &hit13);
GetCtrlVal (multPanel, MULTPNL_3HITNUM, &hit3);
GetCtrlVal (multPanel, MULTPNL_14HITNUM, &hit14);
GetCtrlVal (multPanel, MULTPNL_4HITNUM, &hit4);
GetCtrlVal (multPanel, MULTPNL_15HITNUM, &hit15);
GetCtrlVal (multPanel, MULTPNL_5HITNUM, &hit5);
GetCtrlVal (multPanel, MULTPNL_16HITNUM, &hit16);
GetCtrlVal (multPanel, MULTPNL_6HITNUM, &hit6);
GetCtrlVal (multPanel, MULTPNL_17HITNUM, &hit17);
GetCtrlVal (multPanel, MULTPNL_7HITNUM, &hit7);
GetCtrlVal (multPanel, MULTPNL_18HITNUM, &hit18);
GetCtrlVal (multPanel, MULTPNL_8HITNUM, &hit8);
GetCtrlVal (multPanel, MULTPNL_19HITNUM, &hit19);
GetCtrlVal (multPanel, MULTPNL_9HITNUM, &hit9);
GetCtrlVal (multPanel, MULTPNL_20HITNUM, &hit20);
GetCtrlVal (multPanel, MULTPNL_31HITNUM, &hit31);
GetCtrlVal (multPanel, MULTPNL_21HITNUM, &hit21);
GetCtrlVal (multPanel, MULTPNL_32HITNUM, &hit32);
GetCtrlVal (multPanel, MULTPNL_22HITNUM, &hit22);
GetCtrlVal (multPanel, MULTPNL_33HITNUM, &hit33);
GetCtrlVal (multPanel, MULTPNL_23HITNUM, &hit23);
GetCtrlVal (multPanel, MULTPNL_34HITNUM, &hit34);
GetCtrlVal (multPanel, MULTPNL_24HITNUM, &hit24);
```

```
                GetCtrlVal (multPanel, MULTPNL_35HITNUM, &hit35);
                GetCtrlVal (multPanel, MULTPNL_25HITNUM, &hit25);
                GetCtrlVal (multPanel, MULTPNL_36HITNUM, &hit36);
                GetCtrlVal (multPanel, MULTPNL_26HITNUM, &hit26);
                GetCtrlVal (multPanel, MULTPNL_37HITNUM, &hit37);
                GetCtrlVal (multPanel, MULTPNL_27HITNUM, &hit27);
                GetCtrlVal (multPanel, MULTPNL_38HITNUM, &hit38);
                GetCtrlVal (multPanel, MULTPNL_28HITNUM, &hit28);
                GetCtrlVal (multPanel, MULTPNL_39HITNUM, &hit39);
                GetCtrlVal (multPanel, MULTPNL_29HITNUM, &hit29);
                GetCtrlVal (multPanel, MULTPNL_40HITNUM, &hit40);
                GetCtrlVal (multPanel, MULTPNL_30HITNUM, &hit30);
                GetCtrlVal (multPanel, MULTPNL_41HITNUM, &hit41);
                GetCtrlVal (multPanel, MULTPNL_46HITNUM, &hit46);
                GetCtrlVal (multPanel, MULTPNL_42HITNUM, &hit42);
                GetCtrlVal (multPanel, MULTPNL_47HITNUM, &hit47);
                GetCtrlVal (multPanel, MULTPNL_43HITNUM, &hit43);
                GetCtrlVal (multPanel, MULTPNL_48HITNUM, &hit48);
                GetCtrlVal (multPanel, MULTPNL_44HITNUM, &hit44);
                GetCtrlVal (multPanel, MULTPNL_49HITNUM, &hit49);
                GetCtrlVal (multPanel, MULTPNL_45HITNUM, &hit45);
                GetCtrlVal (multPanel, MULTPNL_50HITNUM, &hit50);
                GetCtrlVal (cnspecPnl, COINPNL_CH0_SW, &ch0en);
                GetCtrlVal (cnspecPnl, COINPNL_CH1_SW, &ch1en);
                GetCtrlVal (cnspecPnl, COINPNL_CH2_SW, &ch2en);
                GetCtrlVal (cnspecPnl, COINPNL_CH3_SW, &ch3en);
                GetCtrlVal (cnspecPnl, COINPNL_CH4_SW, &ch4en);
                GetCtrlVal (cnspecPnl, COINPNL_CH5_SW, &ch5en);
                GetCtrlVal (cnspecPnl, COINPNL_CH6_SW, &ch6en);
                GetCtrlVal (cnspecPnl, COINPNL_CH7_SW, &ch7en);
                GetCtrlVal (cnspecPnl, COINPNL_SHIFTDATA, &response2);

                if(ch0en>0) {
                        histAllocate0(D);
                        SetMenuBarAttribute (menuHandle,
MAIN_MENU_MNUSPEC_MNUSPCH0,
                                ATTR_DIMMED, 0);
                        }
                if(ch1en>0) {
                        histAllocate1(D);
                        SetMenuBarAttribute (menuHandle,
MAIN_MENU_MNUSPEC_MNUSPCH1,
                                ATTR_DIMMED, 0);
                        }
                if(ch2en>0) {
                        histAllocate2(D);
                        SetMenuBarAttribute (menuHandle,
MAIN_MENU_MNUSPEC_MNUSPCH2,
                                ATTR_DIMMED, 0);
                        }
                if(ch3en>0) {
                        histAllocate3(D);
```

```
                        SetMenuBarAttribute (menuHandle,
MAIN_MENU_MNUSPEC_MNUSPCH3,
                                ATTR_DIMMED, 0);
                        }
                if(ch4en>0) {
                        histAllocate4(D);
                        SetMenuBarAttribute (menuHandle,
MAIN_MENU_MNUSPEC_MNUSPCH4,
                                ATTR_DIMMED, 0);
                        }
                if(ch5en>0) {
                        histAllocate5(D);
                        SetMenuBarAttribute (menuHandle,
MAIN_MENU_MNUSPEC_MNUSPCH5,
                                ATTR_DIMMED, 0);
                                }
                if(ch6en>0) {
                        histAllocate6(D);
                        SetMenuBarAttribute (menuHandle,
MAIN_MENU_MNUSPEC_MNUSPCH6,
                                ATTR_DIMMED, 0);
                        }
                if(ch7en>0) {
                        histAllocate7(D);
                        SetMenuBarAttribute (menuHandle,
MAIN_MENU_MNUSPEC_MNUSPCH7,
                                ATTR_DIMMED, 0);
                                }

                GetCtrlVal (cnspecPnl, COINPNL_AR1_SW, &response);
                GetCtrlVal (cnspecPnl, COINPNL_SAVRES1, &response1);
                if(response>0){
                        currTArray=Event1TArray;
                        currDArray=Event1DArray;
                        nHit=hit1;
                        n=1;
                        if(response2>0){
                                shiftData();
                                        }
                        if(response2<1){
                        convertData();
                                }
                            }

                GetCtrlVal (cnspecPnl, COINPNL_AR2_SW, &response);
                GetCtrlVal (cnspecPnl, COINPNL_SAVRES2, &response1);
                if(response>0){
                        currTArray=Event2TArray;
                        currDArray=Event2DArray;
                        nHit=hit2;
                        n=2;
                        if(response2>0){
                                shiftData();
```

```
                                        }
                        if(response2<1){
                        convertData();
                                }
                        }

GetCtrlVal (cnspecPnl, COINPNL_AR3_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES3, &response1);
if(response>0){
                        currTArray=Event3TArray;
                        currDArray=Event3DArray;
                        nHit=hit3;
                        n=3;
                        if(response2>0){
                                shiftData();
                                        }
                        if(response2<1){
                        convertData();
                                }

                                }
GetCtrlVal (cnspecPnl, COINPNL_AR4_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES4, &response1);
if(response>0){
                        currTArray=Event4TArray;
                        currDArray=Event4DArray;
                        nHit=hit4;
                        n=4;
                        if(response2>0){
                                shiftData();
                                        }
                        if(response2<1){
                        convertData();
                                }

                                }
GetCtrlVal (cnspecPnl, COINPNL_AR5_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES5, &response1);
if(response>0){
                        currTArray=Event5TArray;
                        currDArray=Event5DArray;
                        nHit=hit5;
                        n=5;
                        if(response2>0){
                                shiftData();
                                        }
                        if(response2<1){
                        convertData();
                                }

                                }
GetCtrlVal (cnspecPnl, COINPNL_AR6_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES6, &response1);
```

```
if(response>0){
                    currTArray=Event6TArray;
                    currDArray=Event6DArray;
                    nHit=hit6;
                    n=6;
                    if(response2>0){
                              shiftData();
                                        }
                    if(response2<1){
                    convertData();
                              }

                         }
GetCtrlVal (cnspecPnl, COINPNL_AR7_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES7, &response1);
if(response>0){
                    currTArray=Event7TArray;
                    currDArray=Event7DArray;
                    nHit=hit7;
                    n=7;
                    if(response2>0){
                              shiftData();
                                        }
                    if(response2<1){
                    convertData();
                              }

                         }
GetCtrlVal (cnspecPnl, COINPNL_AR8_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES8, &response1);
if(response>0){
                    currTArray=Event8TArray;
                    currDArray=Event8DArray;
                    nHit=hit8;
                    n=8;
                    if(response2>0){
                              shiftData();
                                        }
                    if(response2<1){
                    convertData();
                              }
                        }
GetCtrlVal (cnspecPnl, COINPNL_AR9_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES9, &response1);
if(response>0){
                    currTArray=Event9TArray;
                    currDArray=Event9DArray;
                    nHit=hit9;
                    n=9;
                    if(response2>0){
                              shiftData();
                                        }
                    if(response2<1){
```

```
                convertData();
                        }

                }
GetCtrlVal (cnspecPnl, COINPNL_AR10_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES10, &response1);
if(response>0){
                currTArray=Event10TArray;
                currDArray=Event10DArray;
                nHit=hit10;
                n=10;
                if(response2>0){
                        shiftData();
                                }
                if(response2<1){
                convertData();
                        }

                }
GetCtrlVal (cnspecPnl, COINPNL_AR11_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES11, &response1);
if(response>0){
                currTArray=Event11TArray;
                currDArray=Event11DArray;
                nHit=hit11;
                n=11;
                if(response2>0){
                        shiftData();
                                }
                if(response2<1){
                convertData();
                        }

                }
GetCtrlVal (cnspecPnl, COINPNL_AR12_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES12, &response1);
if(response>0){
                currTArray=Event12TArray;
                currDArray=Event12DArray;
                nHit=hit12;
                n=12;
                if(response2>0){
                        shiftData();
                                }
                if(response2<1){
                convertData();
                        }

                }
GetCtrlVal (cnspecPnl, COINPNL_AR13_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES13, &response1);
if(response>0){
                currTArray=Event13TArray;
```

```
                currDArray=Event13DArray;
                nHit=hit13;
                n=13;
                if(response2>0){
                        shiftData();
                                }
                if(response2<1){
                convertData();
                        }

                    }
GetCtrlVal (cnspecPnl, COINPNL_AR14_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES14, &response1);
if(response>0){
                currTArray=Event14TArray;
                currDArray=Event14DArray;
                nHit=hit14;
                n=14;
                if(response2>0){
                        shiftData();
                                }
                if(response2<1){
                convertData();
                        }

                    }
GetCtrlVal (cnspecPnl, COINPNL_AR15_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES15, &response1);
if(response>0){
                currTArray=Event15TArray;
                currDArray=Event15DArray;
                nHit=hit15;
                n=15;
                if(response2>0){
                        shiftData();
                                }
                if(response2<1){
                convertData();
                        }
                    }
GetCtrlVal (cnspecPnl, COINPNL_AR16_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES16, &response1);
if(response>0){
                currTArray=Event16TArray;
                currDArray=Event16DArray;
                nHit=hit16;
                n=16;
                if(response2>0){
                        shiftData();
                                }
                if(response2<1){
                convertData();
                        }
```

```
                    }
GetCtrlVal (cnspecPnl, COINPNL_AR17_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES17, &response1);
if(response>0){
                    currTArray=Event17TArray;
                    currDArray=Event17DArray;
                    nHit=hit17;
                    n=17;
                    if(response2>0){
                              shiftData();
                                   }
                    if(response2<1){
                    convertData();
                              }

                    }
GetCtrlVal (cnspecPnl, COINPNL_AR18_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES18, &response1);
if(response>0){
                    currTArray=Event18TArray;
                    currDArray=Event18DArray;
                    nHit=hit18;
                    n=18;
                    if(response2>0){
                              shiftData();
                                   }
                    if(response2<1){
                    convertData();
                              }

                    }
GetCtrlVal (cnspecPnl, COINPNL_AR19_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES19, &response1);
if(response>0){
                    currTArray=Event19TArray;
                    currDArray=Event19DArray;
                    nHit=hit19;
                    n=19;
                    if(response2>0){
                              shiftData();
                                   }
                    if(response2<1){
                    convertData();
                              }

                    }
GetCtrlVal (cnspecPnl, COINPNL_AR20_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES20, &response1);
if(response>0){
                    currTArray=Event20TArray;
                    currDArray=Event20DArray;
                    nHit=hit20;
```

```
                    n=20;
                    if(response2>0){
                            shiftData();
                                    }
                    if(response2<1){
                    convertData();
                            }

                        }
GetCtrlVal (cnspecPnl, COINPNL_AR21_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES21, &response1);
if(response>0){
                    currTArray=Event21TArray;
                    currDArray=Event21DArray;
                    nHit=hit21;
                    n=21;
                    if(response2>0){
                            shiftData();
                                    }
                    if(response2<1){
                    convertData();
                            }

                        }
GetCtrlVal (cnspecPnl, COINPNL_AR22_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES22, &response1);
if(response>0){
                    currTArray=Event22TArray;
                    currDArray=Event22DArray;
                    nHit=hit22;
                    n=22;
                    if(response2>0){
                            shiftData();
                                    }
                    if(response2<1){
                    convertData();
                            }

                        }
GetCtrlVal (cnspecPnl, COINPNL_AR23_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES23, &response1);
if(response>0){
                    currTArray=Event23TArray;
                    currDArray=Event23DArray;
                    nHit=hit23;
                    n=23;
                    if(response2>0){
                            shiftData();
                                    }
                    if(response2<1){
                    convertData();
                            }
```

```
                        }
GetCtrlVal (cnspecPnl, COINPNL_AR24_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES24, &response1);
if(response>0){
                        currTArray=Event24TArray;
                        currDArray=Event24DArray;
                        nHit=hit24;
                        n=24;
                        if(response2>0){
                                shiftData();
                                        }
                        if(response2<1){
                        convertData();
                                }

                        }
GetCtrlVal (cnspecPnl, COINPNL_AR25_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES25, &response1);
if(response>0){
                        currTArray=Event25TArray;
                        currDArray=Event25DArray;
                        nHit=hit25;
                        n=25;
                        if(response2>0){
                                shiftData();
                                        }
                        if(response2<1){
                        convertData();
                                }

                        }
GetCtrlVal (cnspecPnl, COINPNL_AR26_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES26, &response1);
if(response>0){
                        currTArray=Event26TArray;
                        currDArray=Event26DArray;
                        nHit=hit26;
                        n=26;
                        if(response2>0){
                                shiftData();
                                        }
                        if(response2<1){
                        convertData();
                                }

                        }
GetCtrlVal (cnspecPnl, COINPNL_AR27_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES27, &response1);
if(response>0){
                        currTArray=Event27TArray;
                        currDArray=Event27DArray;
                        nHit=hit27;
                        n=27;
```

```
                    if(response2>0){
                            shiftData();
                                }
                    if(response2<1){
                    convertData();
                            }

                        }
GetCtrlVal (cnspecPnl, COINPNL_AR28_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES28, &response1);
if(response>0){
                    currTArray=Event28TArray;
                    currDArray=Event28DArray;
                    nHit=hit28;
                    n=28;
                    if(response2>0){
                            shiftData();
                                }
                    if(response2<1){
                    convertData();
                            }
                    }
GetCtrlVal (cnspecPnl, COINPNL_AR29_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES29, &response1);
if(response>0){
                    currTArray=Event29TArray;
                    currDArray=Event29DArray;
                    nHit=hit29;
                    n=29;
                    if(response2>0){
                            shiftData();
                                }
                    if(response2<1){
                    convertData();
                            }

                        }
GetCtrlVal (cnspecPnl, COINPNL_AR30_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES30, &response1);
if(response>0){
                    currTArray=Event30TArray;
                    currDArray=Event30DArray;
                    nHit=hit30;
                    n=30;
                    if(response2>0){
                            shiftData();
                                }
                    if(response2<1){
                    convertData();
                            }

                        }
GetCtrlVal (cnspecPnl, COINPNL_AR31_SW, &response);
```

```
GetCtrlVal (cnspecPnl, COINPNL_SAVRES31, &response1);
if(response>0){
                currTArray=Event31TArray;
                currDArray=Event31DArray;
                nHit=hit31;
                n=31;
                if(response2>0){
                        shiftData();
                                }
                if(response2<1){
                convertData();
                        }

                    }
GetCtrlVal (cnspecPnl, COINPNL_AR32_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES32, &response1);
if(response>0){
                currTArray=Event32TArray;
                currDArray=Event32DArray;
                nHit=hit32;
                n=32;
                if(response2>0){
                        shiftData();
                                }
                if(response2<1){
                convertData();
                        }
                    }
GetCtrlVal (cnspecPnl, COINPNL_AR33_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES33, &response1);
if(response>0){
                currTArray=Event33TArray;
                currDArray=Event33DArray;
                nHit=hit33;
                n=33;
                if(response2>0){
                        shiftData();
                                }
                if(response2<1){
                convertData();
                        }
                    }

GetCtrlVal (cnspecPnl, COINPNL_AR34_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES34, &response1);
if(response>0){
                currTArray=Event34TArray;
                currDArray=Event34DArray;
                nHit=hit34;
                n=34;
                if(response2>0){
                        shiftData();
                                }
```

```
                    if(response2<1){
                    convertData();
                            }
                        }
GetCtrlVal (cnspecPnl, COINPNL_AR35_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES35, &response1);
if(response>0){
                    currTArray=Event35TArray;
                    currDArray=Event35DArray;
                    nHit=hit35;
                    n=35;
                    if(response2>0){
                            shiftData();
                                    }
                    if(response2<1){
                    convertData();
                            }
                        }
GetCtrlVal (cnspecPnl, COINPNL_AR36_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES36, &response1);
if(response>0){
                    currTArray=Event36TArray;
                    currDArray=Event36DArray;
                    nHit=hit36;
                    n=36;
                    if(response2>0){
                            shiftData();
                                    }
                    if(response2<1){
                    convertData();
                            }
                        }
GetCtrlVal (cnspecPnl, COINPNL_AR37_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES37, &response1);
if(response>0){
                    currTArray=Event37TArray;
                    currDArray=Event37DArray;
                    nHit=hit37;
                    n=37;
                    if(response2>0){
                            shiftData();
                                    }
                    if(response2<1){
                    convertData();
                            }
                        }
GetCtrlVal (cnspecPnl, COINPNL_AR38_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES38, &response1);
if(response>0){
                    currTArray=Event38TArray;
                    currDArray=Event38DArray;
                    nHit=hit38;
                    n=38;
```

```
                    if(response2>0){
                            shiftData();
                                }
                    if(response2<1){
                    convertData();
                            }
                        }
GetCtrlVal (cnspecPnl, COINPNL_AR39_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES39, &response1);
if(response>0){
                    currTArray=Event39TArray;
                    currDArray=Event39DArray;
                    nHit=hit39;
                    n=39;
                    if(response2>0){
                            shiftData();
                                }
                    if(response2<1){
                    convertData();
                            }
                        }
GetCtrlVal (cnspecPnl, COINPNL_AR40_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES40, &response1);
if(response>0){
                    currTArray=Event40TArray;
                    currDArray=Event40DArray;
                    nHit=hit40;
                    n=40;
                    if(response2>0){
                            shiftData();
                                }
                    if(response2<1){
                    convertData();
                            }
                        }
GetCtrlVal (cnspecPnl, COINPNL_AR41_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES41, &response1);
if(response>0){
                    currTArray=Event41TArray;
                    currDArray=Event41DArray;
                    nHit=hit41;
                    n=41;
                    if(response2>0){
                            shiftData();
                                }
                    if(response2<1){
                    convertData();
                            }
                        }

GetCtrlVal (cnspecPnl, COINPNL_AR42_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES42, &response1);
if(response>0){
```

```
                currTArray=Event42TArray;
                currDArray=Event42DArray;
                nHit=hit42;
                n=42;
                if(response2>0){
                        shiftData();
                                }
                if(response2<1){
                convertData();
                        }
                }
GetCtrlVal (cnspecPnl, COINPNL_AR43_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES43, &response1);
if(response>0){
                currTArray=Event43TArray;
                currDArray=Event43DArray;
                nHit=hit43;
                n=43;
                if(response2>0){
                        shiftData();
                                }
                if(response2<1){
                convertData();
                        }
                }
GetCtrlVal (cnspecPnl, COINPNL_AR44_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES44, &response1);
if(response>0){
                currTArray=Event44TArray;
                currDArray=Event44DArray;
                nHit=hit44;
                n=44;
                if(response2>0){
                        shiftData();
                                }
                if(response2<1){
                convertData();
                        }
                }
GetCtrlVal (cnspecPnl, COINPNL_AR45_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES45, &response1);
if(response>0){
                currTArray=Event45TArray;
                currDArray=Event45DArray;
                nHit=hit45;
                n=45;
                if(response2>0){
                        shiftData();
                                }
                if(response2<1){
                convertData();
                        }
                }
```

```
GetCtrlVal (cnspecPnl, COINPNL_AR46_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES46, &response1);
if(response>0){
                    currTArray=Event46TArray;
                    currDArray=Event46DArray;
                    nHit=hit46;
                    n=46;
                    if(response2>0){
                              shiftData();
                                        }
                    if(response2<1){
                    convertData();
                              }
                         }
GetCtrlVal (cnspecPnl, COINPNL_AR47_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES47, &response1);
if(response>0){
                    currTArray=Event47TArray;
                    currDArray=Event47DArray;
                    nHit=hit47;
                    n=47;
                    if(response2>0){
                              shiftData();
                                        }
                    if(response2<1){
                    convertData();
                              }
                         }
GetCtrlVal (cnspecPnl, COINPNL_AR48_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES48, &response1);
if(response>0){
                    currTArray=Event48TArray;
                    currDArray=Event48DArray;
                    nHit=hit48;
                    n=48;
                    if(response2>0){
                              shiftData();
                                        }
                    if(response2<1){
                    convertData();
                              }
                         }
GetCtrlVal (cnspecPnl, COINPNL_AR49_SW, &response);
GetCtrlVal (cnspecPnl, COINPNL_SAVRES49, &response1);
if(response>0){
                    currTArray=Event49TArray;
                    currDArray=Event49DArray;
                    nHit=hit49;
                    n=49;
                    if(response2>0){
                              shiftData();
                                        }
                    if(response2<1){
```

```
                                        convertData();
                                                }
                                        }
                                GetCtrlVal (cnspecPnl, COINPNL_AR50_SW, &response);
                                GetCtrlVal (cnspecPnl, COINPNL_SAVRES50, &response1);
                                if(response>0){
                                        currTArray=Event50TArray;
                                        currDArray=Event50DArray;
                                        nHit=hit50;
                                        n=50;
                                        if(response2>0){
                                                shiftData();
                                                        }
                                        if(response2<1){
                                        convertData();
                                                }
                                                        }

        HidePanel(cnspecPnl);

                                break;
                        }
                return 0;
}

int CVICALLBACK SaveAllSearchesCB (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event)
                {
                case EVENT_COMMIT:
                        GetCtrlVal (cnspecPnl, COINPNL_SAVRES1, &response);
                        if(response>0)



                                break;
                        }
                return 0;
}

int CVICALLBACK EnableAllArraysCB (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event)
                {
                case EVENT_COMMIT:
                        GetCtrlVal (cnspecPnl, COINPNL_AR1_SW, &response);
```

```
                        break;
                }
        return 0;
}

int CVICALLBACK coincidenceResetCB (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
int x =0;
        switch (event)
                {
                case EVENT_COMMIT:

                DefaultCtrl (multPanel, MULTPNL_1HITNUM);
                DefaultCtrl (multPanel, MULTPNL_2HITNUM);
                DefaultCtrl (multPanel, MULTPNL_3HITNUM);
                DefaultCtrl (multPanel, MULTPNL_4HITNUM);
                DefaultCtrl (multPanel, MULTPNL_5HITNUM);
                DefaultCtrl (multPanel, MULTPNL_6HITNUM);
                DefaultCtrl (multPanel, MULTPNL_7HITNUM);
                DefaultCtrl (multPanel, MULTPNL_8HITNUM);
                DefaultCtrl (multPanel, MULTPNL_9HITNUM);
                DefaultCtrl (multPanel, MULTPNL_10HITNUM);
                DefaultCtrl (multPanel, MULTPNL_11HITNUM);
                DefaultCtrl (multPanel, MULTPNL_12HITNUM);
                DefaultCtrl (multPanel, MULTPNL_13HITNUM);
                DefaultCtrl (multPanel, MULTPNL_14HITNUM);
                DefaultCtrl (multPanel, MULTPNL_15HITNUM);
                DefaultCtrl (multPanel, MULTPNL_16HITNUM);
                DefaultCtrl (multPanel, MULTPNL_17HITNUM);
                DefaultCtrl (multPanel, MULTPNL_18HITNUM);
                DefaultCtrl (multPanel, MULTPNL_19HITNUM);
                DefaultCtrl (multPanel, MULTPNL_20HITNUM);
                DefaultCtrl (multPanel, MULTPNL_21HITNUM);
                DefaultCtrl (multPanel, MULTPNL_22HITNUM);
                DefaultCtrl (multPanel, MULTPNL_23HITNUM);
                DefaultCtrl (multPanel, MULTPNL_24HITNUM);
                DefaultCtrl (multPanel, MULTPNL_25HITNUM);
                DefaultCtrl (multPanel, MULTPNL_26HITNUM);
                DefaultCtrl (multPanel, MULTPNL_27HITNUM);
                DefaultCtrl (multPanel, MULTPNL_28HITNUM);
                DefaultCtrl (multPanel, MULTPNL_29HITNUM);
                DefaultCtrl (multPanel, MULTPNL_30HITNUM);
                DefaultCtrl (multPanel, MULTPNL_31HITNUM);
                DefaultCtrl (multPanel, MULTPNL_32HITNUM);
                DefaultCtrl (multPanel, MULTPNL_33HITNUM);
                DefaultCtrl (multPanel, MULTPNL_34HITNUM);
                DefaultCtrl (multPanel, MULTPNL_35HITNUM);
                DefaultCtrl (multPanel, MULTPNL_36HITNUM);
                DefaultCtrl (multPanel, MULTPNL_37HITNUM);
                DefaultCtrl (multPanel, MULTPNL_38HITNUM);
                DefaultCtrl (multPanel, MULTPNL_39HITNUM);
                DefaultCtrl (multPanel, MULTPNL_40HITNUM);
```

```
DefaultCtrl (multPanel, MULTPNL_41HITNUM);
DefaultCtrl (multPanel, MULTPNL_42HITNUM);
DefaultCtrl (multPanel, MULTPNL_43HITNUM);
DefaultCtrl (multPanel, MULTPNL_44HITNUM);
DefaultCtrl (multPanel, MULTPNL_45HITNUM);
DefaultCtrl (multPanel, MULTPNL_46HITNUM);
DefaultCtrl (multPanel, MULTPNL_47HITNUM);
DefaultCtrl (multPanel, MULTPNL_48HITNUM);
DefaultCtrl (multPanel, MULTPNL_49HITNUM);
DefaultCtrl (multPanel, MULTPNL_50HITNUM);

DefaultCtrl (multPanel, MULTPNL_TOTNUM);
DefaultCtrl (multPanel, MULTPNL_NULLNUM);
SetCtrlVal (cnspecPnl, COINPNL_AR1_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR2_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR3_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR4_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR5_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR6_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR7_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR8_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR9_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR10_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR11_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR12_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR13_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR14_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR15_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR16_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR17_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR18_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR19_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR20_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR21_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR22_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR23_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR24_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR25_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR26_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR27_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR28_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR29_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR30_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR31_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR32_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR33_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR34_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR35_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR36_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR37_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR38_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR39_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR40_SW, 0);
```

```
SetCtrlVal (cnspecPnl, COINPNL_AR41_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR42_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR43_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR44_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR45_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR46_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR47_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR48_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR49_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_AR50_SW, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES1, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES2, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES3, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES4, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES5, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES6, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES7, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES8, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES9, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES10, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES11, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES12, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES13, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES14, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES15, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES16, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES17, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES18, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES19, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES20, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES21, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES22, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES23, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES24, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES25, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES26, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES27, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES28, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES29, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES30, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES31, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES32, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES33, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES34, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES35, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES36, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES37, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES38, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES39, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES40, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES41, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES42, 0);
SetCtrlVal (cnspecPnl, COINPNL_SAVRES43, 0);
```

```
                SetCtrlVal (cnspecPnl, COINPNL_SAVRES44, 0);
                SetCtrlVal (cnspecPnl, COINPNL_SAVRES45, 0);
                SetCtrlVal (cnspecPnl, COINPNL_SAVRES46, 0);
                SetCtrlVal (cnspecPnl, COINPNL_SAVRES47, 0);
                SetCtrlVal (cnspecPnl, COINPNL_SAVRES48, 0);
                SetCtrlVal (cnspecPnl, COINPNL_SAVRES49, 0);
                SetCtrlVal (cnspecPnl, COINPNL_SAVRES50, 0);
                hit1=0; hit2=0; hit3=0; hit4=0; hit5=0; hit6=0; hit7=0; hit8=0; hit9=0; hit10=0;
                hit11=0; hit12=0; hit13=0; hit14=0; hit15=0; hit16=0; hit17=0; hit18=0; hit19=0;
hit20=0;
                hit21=0; hit22=0; hit23=0; hit24=0; hit25=0; hit26=0; hit27=0; hit28=0; hit29=0;
hit30=0;
                hit31=0; hit32=0; hit33=0; hit34=0; hit35=0; hit36=0; hit37=0; hit38=0; hit39=0;
hit40=0;
                hit41=0; hit42=0; hit43=0; hit44=0; hit45=0; hit46=0; hit47=0; hit48=0; hit49=0;
hit50=0;
                totalEvents=0; nullEvents=0; bigEvents = 0;
                stop1Events=0; stop2Events=0; stop3Events=0; stop4Events=0; stop5Events = 0;
                stop6Events=0; stop7Events=0; stop8Events=0; stop9Events=0; stop10Events = 0;
                stop11Events=0; stop12Events=0; stop13Events=0; stop14Events=0; stop15Events = 0;
                stop16Events=0; stop17Events=0; stop18Events=0; stop19Events=0; stop20Events = 0;
                stop21Events=0; stop22Events=0; stop23Events=0; stop24Events=0; stop25Events = 0;
                stop26Events=0; stop27Events=0; stop28Events=0; stop29Events=0; stop30Events = 0;
                stop31Events=0; stop32Events=0; stop33Events=0; stop34Events=0; stop35Events = 0;
                stop36Events=0; stop37Events=0; stop38Events=0; stop39Events=0; stop40Events = 0;
                stop41Events=0; stop42Events=0; stop43Events=0; stop44Events=0; stop45Events = 0;
                stop46Events=0; stop47Events=0; stop48Events=0; stop49Events=0; stop50Events = 0;
                ea1=0;ea2=0;ea3=0;ea4=0;ea5=0;ea6=0;ea7=0;ea8=0;ea9=0;ea10=0;
                ea11=0;ea12=0;ea13=0;ea14=0;ea15=0;ea16=0;ea17=0;ea18=0;ea19=0;ea20=0;
                ea21=0;ea22=0;ea23=0;ea24=0;ea25=0;ea26=0;ea27=0;ea28=0;ea29=0;ea30=0;
                ea31=0;ea32=0;ea33=0;ea34=0;ea35=0;ea36=0;ea37=0;ea38=0;ea39=0;ea40=0;
                ea41=0;ea42=0;ea43=0;ea44=0;ea45=0;ea46=0;ea47=0;ea48=0;ea49=0;ea50=0;
                if(Event1TArray!=NULL){free(Event1TArray);Event1TArray=NULL;
                free(Event1DArray);Event1DArray=NULL;};
                if(Event2TArray!=NULL){free(Event2TArray);Event2TArray=NULL;
                free(Event2DArray);Event2DArray=NULL;};
                if(Event3TArray!=NULL){free(Event3TArray);Event3TArray=NULL;
                free(Event3DArray);Event3DArray=NULL;};
                if(Event4TArray!=NULL){free(Event4TArray);Event4TArray=NULL;
                free(Event4DArray);Event4DArray=NULL;};
                if(Event5TArray!=NULL){free(Event5TArray);Event5TArray=NULL;
                free(Event5DArray);Event5DArray=NULL;};
                if(Event6TArray!=NULL){free(Event6TArray);Event6TArray=NULL;
                free(Event6DArray);Event6DArray=NULL;};
                if(Event7TArray!=NULL){free(Event7TArray);Event7TArray=NULL;
                free(Event7DArray);Event7DArray=NULL;};
                if(Event8TArray!=NULL){free(Event8TArray);Event8TArray=NULL;
                free(Event8DArray);Event8DArray=NULL;};
                if(Event9TArray!=NULL){free(Event9TArray);Event9TArray=NULL;
                free(Event9DArray);Event9DArray=NULL;};
                if(Event10TArray!=NULL){free(Event10TArray);Event10TArray=NULL;
                free(Event10DArray);Event10DArray=NULL;};
                if(Event11TArray!=NULL){free(Event11TArray);Event11TArray=NULL;
```

```
free(Event11DArray);Event11DArray=NULL;};
if(Event12TArray!=NULL){free(Event12TArray);Event12TArray=NULL;
free(Event12DArray);Event12DArray=NULL;};
if(Event13TArray!=NULL){free(Event13TArray);Event13TArray=NULL;
free(Event13DArray);Event13DArray=NULL;};
if(Event14TArray!=NULL){free(Event14TArray);Event14TArray=NULL;
free(Event14DArray);Event14DArray=NULL;};
if(Event15TArray!=NULL){free(Event15TArray);Event15TArray=NULL;
free(Event15DArray);Event15DArray=NULL;};
if(Event16TArray!=NULL){free(Event16TArray);Event16TArray=NULL;
free(Event16DArray);Event16DArray=NULL;};
if(Event17TArray!=NULL){free(Event17TArray);Event17TArray=NULL;
free(Event17DArray);Event17DArray=NULL;};
if(Event18TArray!=NULL){free(Event18TArray);Event18TArray=NULL;
free(Event18DArray);Event18DArray=NULL;};
if(Event19TArray!=NULL){free(Event19TArray);Event19TArray=NULL;
free(Event19DArray);Event19DArray=NULL;};
if(Event20TArray!=NULL){free(Event20TArray);Event20TArray=NULL;
free(Event20DArray);Event20DArray=NULL;};
if(Event21TArray!=NULL){free(Event21TArray);Event21TArray=NULL;
free(Event11DArray);Event21DArray=NULL;};
if(Event22TArray!=NULL){free(Event22TArray);Event22TArray=NULL;
free(Event22DArray);Event22DArray=NULL;};
if(Event23TArray!=NULL){free(Event23TArray);Event23TArray=NULL;
free(Event23DArray);Event23DArray=NULL;};
if(Event24TArray!=NULL){free(Event24TArray);Event24TArray=NULL;
free(Event24DArray);Event24DArray=NULL;};
if(Event25TArray!=NULL){free(Event25TArray);Event25TArray=NULL;
free(Event25DArray);Event25DArray=NULL;};
if(Event26TArray!=NULL){free(Event26TArray);Event26TArray=NULL;
free(Event26DArray);Event26DArray=NULL;};
if(Event27TArray!=NULL){free(Event27TArray);Event27TArray=NULL;
free(Event27DArray);Event27DArray=NULL;};
if(Event28TArray!=NULL){free(Event28TArray);Event28TArray=NULL;
free(Event28DArray);Event28DArray=NULL;};
if(Event29TArray!=NULL){free(Event29TArray);Event29TArray=NULL;
free(Event29DArray);Event29DArray=NULL;};
if(Event30TArray!=NULL){free(Event30TArray);Event30TArray=NULL;
free(Event30DArray);Event30DArray=NULL;};
if(Event31TArray!=NULL){free(Event31TArray);Event31TArray=NULL;
free(Event31DArray);Event31DArray=NULL;};
if(Event32TArray!=NULL){free(Event32TArray);Event32TArray=NULL;
free(Event32DArray);Event32DArray=NULL;};
if(Event33TArray!=NULL){free(Event33TArray);Event33TArray=NULL;
free(Event33DArray);Event33DArray=NULL;};
if(Event34TArray!=NULL){free(Event34TArray);Event34TArray=NULL;
free(Event34DArray);Event34DArray=NULL;};
if(Event35TArray!=NULL){free(Event35TArray);Event35TArray=NULL;
free(Event35DArray);Event35DArray=NULL;};
if(Event36TArray!=NULL){free(Event36TArray);Event36TArray=NULL;
free(Event36DArray);Event36DArray=NULL;};
if(Event37TArray!=NULL){free(Event37TArray);Event37TArray=NULL;
free(Event37DArray);Event37DArray=NULL;};
```

```
                    if(Event38TArray!=NULL){free(Event38TArray);Event38TArray=NULL;
                    free(Event38DArray);Event38DArray=NULL;};
                    if(Event39TArray!=NULL){free(Event39TArray);Event39TArray=NULL;
                    free(Event39DArray);Event39DArray=NULL;};
                    if(Event40TArray!=NULL){free(Event40TArray);Event40TArray=NULL;
                    free(Event40DArray);Event40DArray=NULL;};
                    if(Event41TArray!=NULL){free(Event41TArray);Event41TArray=NULL;
                    free(Event41DArray);Event41DArray=NULL;};
                    if(Event42TArray!=NULL){free(Event42TArray);Event42TArray=NULL;
                    free(Event42DArray);Event42DArray=NULL;};
                    if(Event43TArray!=NULL){free(Event43TArray);Event43TArray=NULL;
                    free(Event43DArray);Event43DArray=NULL;};
                    if(Event44TArray!=NULL){free(Event44TArray);Event44TArray=NULL;
                    free(Event44DArray);Event44DArray=NULL;};
                    if(Event45TArray!=NULL){free(Event45TArray);Event45TArray=NULL;
                    free(Event45DArray);Event45DArray=NULL;};
                    if(Event46TArray!=NULL){free(Event46TArray);Event46TArray=NULL;
                    free(Event46DArray);Event46DArray=NULL;};
                    if(Event47TArray!=NULL){free(Event47TArray);Event47TArray=NULL;
                    free(Event47DArray);Event47DArray=NULL;};
                    if(Event48TArray!=NULL){free(Event48TArray);Event48TArray=NULL;
                    free(Event48DArray);Event48DArray=NULL;};
                    if(Event49TArray!=NULL){free(Event49TArray);Event49TArray=NULL;
                    free(Event49DArray);Event49DArray=NULL;};
                    if(Event50TArray!=NULL){free(Event50TArray);Event50TArray=NULL;
                    free(Event50DArray);Event50DArray=NULL;};
                    if (array!=NULL) {free(array);array=NULL;}

                            break;
                    }
            return 0;
}

int CVICALLBACK ExitCoincidenceCB (int panel, int control, int event,
                    void *callbackData, int eventData1, int eventData2)
{
            switch (event)
                    {
                    case EVENT_COMMIT:


                            HidePanel(cnspecPnl);

                            break;
                    }
            return 0;
}

/*****************************************Multiplicity*****************************
*************************************************************/
void CVICALLBACK MenuMultiplicityReportCB (int menuBar, int menuItem, void *callbackData,
                    int panel)
{
```

```
	DisplayPanel (multPanel);
}

int CVICALLBACK HideMultiplicityReport (int panel, int control, int event,
		void *callbackData, int eventData1, int eventData2)
{
	switch (event)
		{
		case EVENT_COMMIT:
				HidePanel (multPanel);
			break;
		}
	return 0;
}

int CVICALLBACK MultDistCB (int panel, int control, int event,
		void *callbackData, int eventData1, int eventData2)
{
	switch (event)
		{
		case EVENT_COMMIT:

			GetCtrlVal (multPanel, MULTPNL_1HITNUM, &hit1);
			GetCtrlVal (multPanel, MULTPNL_2HITNUM, &hit2);
			GetCtrlVal (multPanel, MULTPNL_3HITNUM, &hit3);
			GetCtrlVal (multPanel, MULTPNL_4HITNUM, &hit4);
			GetCtrlVal (multPanel, MULTPNL_5HITNUM, &hit5);
			GetCtrlVal (multPanel, MULTPNL_6HITNUM, &hit6);
			GetCtrlVal (multPanel, MULTPNL_7HITNUM, &hit7);
			GetCtrlVal (multPanel, MULTPNL_8HITNUM, &hit8);
			GetCtrlVal (multPanel, MULTPNL_9HITNUM, &hit9);
			GetCtrlVal (multPanel, MULTPNL_10HITNUM, &hit10);
			GetCtrlVal (multPanel, MULTPNL_11HITNUM, &hit11);
			GetCtrlVal (multPanel, MULTPNL_12HITNUM, &hit12);
			GetCtrlVal (multPanel, MULTPNL_13HITNUM, &hit13);
			GetCtrlVal (multPanel, MULTPNL_14HITNUM, &hit14);
			GetCtrlVal (multPanel, MULTPNL_15HITNUM, &hit15);
			GetCtrlVal (multPanel, MULTPNL_16HITNUM, &hit16);
			GetCtrlVal (multPanel, MULTPNL_17HITNUM, &hit17);
			GetCtrlVal (multPanel, MULTPNL_18HITNUM, &hit18);
			GetCtrlVal (multPanel, MULTPNL_19HITNUM, &hit19);
			GetCtrlVal (multPanel, MULTPNL_20HITNUM, &hit20);
			GetCtrlVal (multPanel, MULTPNL_21HITNUM, &hit21);
			GetCtrlVal (multPanel, MULTPNL_22HITNUM, &hit22);
			GetCtrlVal (multPanel, MULTPNL_23HITNUM, &hit23);
			GetCtrlVal (multPanel, MULTPNL_24HITNUM, &hit24);
			GetCtrlVal (multPanel, MULTPNL_25HITNUM, &hit25);
			GetCtrlVal (multPanel, MULTPNL_26HITNUM, &hit26);
			GetCtrlVal (multPanel, MULTPNL_27HITNUM, &hit27);
			GetCtrlVal (multPanel, MULTPNL_28HITNUM, &hit28);
			GetCtrlVal (multPanel, MULTPNL_29HITNUM, &hit29);
			GetCtrlVal (multPanel, MULTPNL_30HITNUM, &hit30);
			GetCtrlVal (multPanel, MULTPNL_31HITNUM, &hit31);
```

```
GetCtrlVal (multPanel, MULTPNL_32HITNUM, &hit32);
GetCtrlVal (multPanel, MULTPNL_33HITNUM, &hit33);
GetCtrlVal (multPanel, MULTPNL_34HITNUM, &hit34);
GetCtrlVal (multPanel, MULTPNL_35HITNUM, &hit35);
GetCtrlVal (multPanel, MULTPNL_36HITNUM, &hit36);
GetCtrlVal (multPanel, MULTPNL_37HITNUM, &hit37);
GetCtrlVal (multPanel, MULTPNL_38HITNUM, &hit38);
GetCtrlVal (multPanel, MULTPNL_39HITNUM, &hit39);
GetCtrlVal (multPanel, MULTPNL_40HITNUM, &hit40);
GetCtrlVal (multPanel, MULTPNL_41HITNUM, &hit41);
GetCtrlVal (multPanel, MULTPNL_42HITNUM, &hit42);
GetCtrlVal (multPanel, MULTPNL_43HITNUM, &hit43);
GetCtrlVal (multPanel, MULTPNL_44HITNUM, &hit44);
GetCtrlVal (multPanel, MULTPNL_45HITNUM, &hit45);
GetCtrlVal (multPanel, MULTPNL_46HITNUM, &hit46);
GetCtrlVal (multPanel, MULTPNL_47HITNUM, &hit47);
GetCtrlVal (multPanel, MULTPNL_48HITNUM, &hit48);
GetCtrlVal (multPanel, MULTPNL_49HITNUM, &hit49);
GetCtrlVal (multPanel, MULTPNL_50HITNUM, &hit50);
GetCtrlVal (multPanel, MULTPNL_NULLNUM, &nullEvents);

MultArray[0] = nullEvents;
MultArray[1] = hit1; MultArray[2] = hit2; MultArray[3] = hit3;
MultArray[4] = hit4;
MultArray[5] = hit5; MultArray[6] = hit6; MultArray[7] = hit7;
MultArray[8] = hit8;
MultArray[9] = hit9; MultArray[10] = hit10; MultArray[11] = hit11;
MultArray[12] = hit12;
MultArray[13] = hit13; MultArray[14] = hit14; MultArray[15] = hit15;
MultArray[16] = hit16;
MultArray[17] = hit17; MultArray[18] = hit18; MultArray[19] = hit19;
MultArray[20] = hit20;
MultArray[21] = hit21; MultArray[22] = hit22; MultArray[23] = hit23;
MultArray[24] = hit24;
MultArray[25] = hit25; MultArray[26] = hit26; MultArray[27] = hit27;
MultArray[28] = hit28;
MultArray[29] = hit29; MultArray[30] = hit30; MultArray[31] = hit31;
MultArray[32] = hit32;
MultArray[33] = hit33; MultArray[34] = hit34; MultArray[35] = hit35;
MultArray[36] = hit36;
MultArray[37] = hit37; MultArray[38] = hit38; MultArray[39] = hit39;
MultArray[40] = hit40;
MultArray[41] = hit41; MultArray[42] = hit42; MultArray[43] = hit43;
MultArray[44] = hit44;
MultArray[45] = hit45; MultArray[46] = hit46; MultArray[47] = hit47;
MultArray[48] = hit48;
MultArray[49] = hit49; MultArray[50] = hit50;

for(x=0; x<=50; x++){

        MultXArray[x] = x;

                }
```

```
                        currHist = MultArray;
                        DeleteGraphPlot (mnPanel, MN_PANEL_GRAPH, -1,
VAL_IMMEDIATE_DRAW);
                        RefreshGraph (mnPanel, MN_PANEL_GRAPH);
                        strcpy(buf,"# Secondary Ions Emitted/Event");
                        SetCtrlAttribute (mnPanel, MN_PANEL_GRAPH, ATTR_XNAME, buf);
                        strcpy(buf,"#SI/Event * n");
                        SetCtrlAttribute (mnPanel, MN_PANEL_GRAPH, ATTR_YNAME, buf);
                        PlotXY (mnPanel, MN_PANEL_GRAPH, MultXArray, currHist, 51,
                                VAL_UNSIGNED_INTEGER,
VAL_UNSIGNED_INTEGER,
                                VAL_BASE_ZERO_VERTICAL_BAR,
VAL_EMPTY_SQUARE, VAL_SOLID, 1,
                                VAL_RED);

                SetCtrlAttribute (mnPanel, MN_PANEL_GRAPH, ATTR_XPRECISION, 0);

                        break;

                }
        return 0;
}
/*****************************************ResetFunction*****************************
***/
int CVICALLBACK ResetCB (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event)
                {
                int i = 0;

                case EVENT_COMMIT:

iStatus = 0;
iRetVal = 0;
iDevice = 1;
iGroup = 1;
iPort = 0;
iDir = 0;
iSignal = 1;
iEdge = 0;
iAckDelayTime =1;
iDBModeON = 1;
iDBModeOFF = 0;
iOldDataStop = 1;
iPartialTransfer = 0;
iHalfReady = 0;
iLoopCount = 0;
iHalfBufsToRead = C;
ulAlignIndex = 0;
iResource = 11;
iIgnoreWarning = 1;
lTimeout = 180;
```

```
iYieldON = 1;

iGroupSize = 2;
iReqPol = 1;
iAckPol = 1;
ulCount = E;
ulPtsTfr = (E/2);
ulBufferSize = E;

                    DefaultCtrl (masscalibPnl, CALIBPNL_NUMCH1);
                    DefaultCtrl (masscalibPnl, CALIBPNL_NUMCH2);
                    DefaultCtrl (masscalibPnl, CALIBPNL_NUMCH3);
                    DefaultCtrl (masscalibPnl, CALIBPNL_NUMCH4);
                    DefaultCtrl (masscalibPnl, CALIBPNL_NUMCH5);
                    DefaultCtrl (masscalibPnl, CALIBPNL_NUMCH6);
                    DefaultCtrl (masscalibPnl, CALIBPNL_NUMCH7);
                    DefaultCtrl (masscalibPnl, CALIBPNL_NUMCH8);
                    DefaultCtrl (masscalibPnl, CALIBPNL_NUMMASS1);
                    DefaultCtrl (masscalibPnl, CALIBPNL_NUMMASS2);
                    DefaultCtrl (masscalibPnl, CALIBPNL_NUMMASS3);
                    DefaultCtrl (masscalibPnl, CALIBPNL_NUMMASS4);
                    DefaultCtrl (masscalibPnl, CALIBPNL_NUMMASS5);
                    DefaultCtrl (masscalibPnl, CALIBPNL_NUMMASS6);
                    DefaultCtrl (masscalibPnl, CALIBPNL_NUMMASS7);
                    DefaultCtrl (masscalibPnl, CALIBPNL_NUMMASS8);
                    DefaultCtrl (masscalibPnl, CALIBPNL_NUMERR);
                    DefaultCtrl (masscalibPnl, CALIBPNL_NUMINTER);
                    DefaultCtrl (masscalibPnl, CALIBPNL_NUMSLOPE);
                    DefaultCtrl (cnspecPnl, COINPNL_NUMCOUP);
                    DefaultCtrl (cnspecPnl, COINPNL_NUMCOLO);
                    DefaultCtrl (mnPanel, MN_PANEL_NUMPKAREA);
                    DefaultCtrl (mnPanel, MN_PANEL_NUMYREF);
                    DefaultCtrl (mnPanel, MN_PANEL_NUMXREF);
                    DefaultCtrl (mnPanel, MN_PANEL_NUMEVENT);
                    DefaultCtrl (expPnl, EXP_PNL_OPTEXTBOX);
                    DefaultCtrl (expPnl, EXP_PNL_IDTEXTBOX);
                    DefaultCtrl (expPnl, EXP_PNL_TRTEXTBOX);
                    DefaultCtrl (expPnl, EXP_PNL_PITEXTBOX);
                    DefaultCtrl (expPnl, EXP_PNL_SIKTXTBOX);
                    DefaultCtrl (expPnl, EXP_PNL_STDTXTBOX);
                    DefaultCtrl (expPnl, EXP_PNL_ST1TXTBOX);
                    DefaultCtrl (expPnl, EXP_PNL_ST2TXTBOX);
                    DefaultCtrl (expPnl, EXP_PNL_MAXCHAN);
                    DefaultCtrl (expPnl, EXP_PNL_DT1TXTBOX);
                    DefaultCtrl (expPnl, EXP_PNL_TM1TXTBOX);
                    DefaultCtrl (expPnl, EXP_PNL_DT2TXTBOX);
                    DefaultCtrl (expPnl, EXP_PNL_TM2TXTBOX);
                    strcpy(buf,"Channel Number");
                    SetCtrlAttribute (mnPanel, MN_PANEL_GRAPH, ATTR_XNAME, buf);
                    strcpy(buf,"Intensity");
                    SetCtrlAttribute (mnPanel, MN_PANEL_GRAPH, ATTR_YNAME, buf);
                    SetCtrlAttribute (mnPanel, MN_PANEL_GRAPH, ATTR_XPRECISION, 0);
                    SetCtrlAttribute (mnPanel, MN_PANEL_GRAPH, ATTR_YPRECISION, 0);
```

```
SetCtrlAttribute (mnPanel, MN_PANEL_NUMYREF, ATTR_PRECISION, 0);
response = 0;
response1 = 0;

if(coinspec0!=NULL){
        free(coinspec0);
        coinspec0=NULL;
            }
if(coinspec1!=NULL){
        free(coinspec1);
        coinspec1=NULL;
            }
if(coinspec2!=NULL){
        free(coinspec2);
        coinspec2=NULL;
            }
if(coinspec3!=NULL){
        free(coinspec3);
        coinspec3=NULL;
            }
if(coinspec4!=NULL){
        free(coinspec4);
        coinspec4=NULL;
            }
if(coinspec5!=NULL){
        free(coinspec5);
        coinspec5=NULL;
            }
if(coinspec6!=NULL){
        free(coinspec6);
        coinspec6=NULL;
            }
if(coinspec7!=NULL){
        free(coinspec7);
        coinspec7=NULL;
            }
if (array!=NULL) {
        free(array);
        array=NULL;
              }
if (coinspec!=NULL) {
        free(coinspec);
        coinspec=NULL;
              }
if (piBuffer!=NULL) {
        free(piBuffer);
              }
if (piHalfBuffer!=NULL) {
        free(piHalfBuffer);
        piHalfBuffer=NULL;
              }
if (Mass!=NULL) {
        free(Mass);
```

```
                    Mass=NULL;
                        }
                if (normArray!=NULL) {
                    free(normArray);
                    normArray=NULL;
                        }
                i=0;
                for(i=0;i<=7;i++){

                    MassCalibX[i] = 0;
                    MassCalibY[i] = 0;
                    MassCalib[i] = 0;

                                                }
                i=0;
                for(i=0;i<=50;i++){

                    MultArray[i] = 0;

                                                }
        currHist=NULL;
        DeleteGraphPlot (mnPanel, MN_PANEL_GRAPH, -1, VAL_IMMEDIATE_DRAW);
        RefreshGraph (mnPanel, MN_PANEL_GRAPH);
        size = 0;
                    break;
                }
        return 0;
}
```

VITA

Richard Dale Rickman was born in Memphis, Tennessee on December 31, 1962. He is the son of Robert and Paula Rickman. He graduated from Delta C-7 High School in Deering, Missouri in May of 1981. After graduation he enlisted in the United States Air Force attaining the rank of Staff Sergeant before being Honorably Discharged in 1989. He was a driver/trainer of J. B. Hunt Transport for 1990 to 1995. He returned to school in 1996 and received his B. S. in Chemistry from Arkansas State University. In August 1999 he enrolled in Texas A&M University to pursue a Ph.D. degree in surface mass spectrometry under the guidance of Dr. Emile A. Schweikert.

His permanent address is that of his parents:

406 Walter Street

Kennett, Missouri 63857